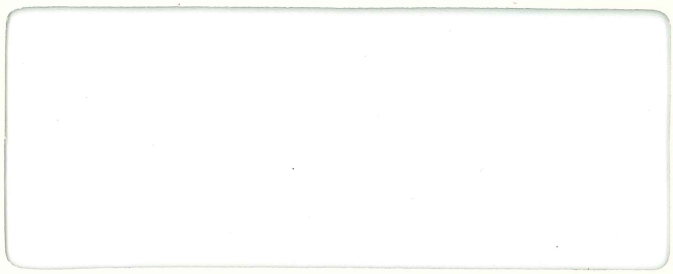
DEC Standard Runoff (DSR) User's Guide

Order No. AA-J268A-TK

Software Tools

digital



DEC Standard Runoff
(DSR) User's Guide

Order No. AA-J268A-TK

November 1979

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard, massachusetts

First Printing, November 1979

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10
VAX	VMS	SBI
DECnet	IAS	PDT
DATATRIEVE	TRAX	

DEC STANDARD RUNOFF(DSR) PROGRAM

USER MANUAL

CHAPTER 3 THE BASIC DSR COMMANDS

ILLUSTRATIONS

TABLES

PREFACE

The first two chapters of this manual introduce the concept of automated text formatting via the DEC STANDARD RUNOFF (DSR) Text Formatting Program. The material then provides to both the inexperienced and advanced user of text processing systems the information required to successfully format a wide range of documents.

- Chapter 1 provides the inexperienced user with an overview of automated text formatting, describing the preparation, processing, and final production of a document. This is followed by an introduction to the DSR program.
- CHAPTER 2 describes the elements of the DSR language (i.e., commands and flags), including basic rules and conventions.
- CHAPTER 3 describes all DSR commands and usage, with the exception if index and table of contents commands.
- CHAPTER 4 describes the DSR flags, usage, and related control commands.
- CHAPTER 5 provides the information required to create an index.
- CHAPTER 6 provides the information required to create a table of contents.
- CHAPTER 7 provides system-related (i.e., TOPS-10, TOPS-20, VMS, etc.) descriptions of the DSR command line switches and their use.
- APPENDIX-A provides descriptions of certain DSR features that are included to provide compatibility with earlier versions of RUNOFF.
- APPENDIX-B provides a listing of the DSR messages (error/non-error).
- APPENDIX-C provides an alphabetic listing of all DSR commands.
- APPENDIX-D provides examples of the formatting of templates.

Although much of the material contained herein is directed toward the the needs of the inexperienced user, it is not within the scope of this manual to describe the use of input terminals, system monitors, or text editor programs. If such information is required, the reader should refer to applicable user documentation.

CHAPTER 1

INTRODUCTION

1.1 AUTOMATED TEXT PROCESSING

Automated text processing is a three-step process for producing automatically edited and formatted documents from initial draft copy.

During the first step, a text editor (e.g. SOS) is used to create or access a source (input) file for the insertion and editing of the rough draft. During the second step, a document formatter (e.g. RUNOFF) is used to derive a satisfactorily formatted object (output) file, from the input file, for the third and final step: the printing and distribution of the document.

Document Formatter Program

The principal benefits of a formatter, such as the DSR, are in the speed and ease with which complex material may be processed: from rough draft copy, through corrective cycles, and into final production.

Using formatter commands and flags (special characters), in the input file, complex documents may be optionally titled, paged, and sectioned. An index may be added, and the subject matter may be enhanced with bolding, underlining, or overstriking. When a final draft version is derived for printing, none of the commands or flags used to shape the text appear in the output file.

Following each correction cycle, the formatter can immediately be used to produce a new version of the document reflecting the latest changes.

1.1.1 Document Processing

During document preparation and processing, the user:

1. Creates an input file (or edits an existent file) for the insertion of text, along with commands and flags that specify the formatting of the document.

2. Submits the input file to the document formatter to generate a draft version (output file) that contains the current changes.

3. Depending on the appearance or current status of the draft version, the user either:

   (a) returns to Step 1 for corrective action, or

   (b) sends the document to the printer.

1.1.2 Document Production

During document production, the final copies of the document are printed for distribution. Presently, line printers, or the high quality printing afforded by a DIABLO Printer, are the most common methods of production.

1.2 DEC STANDARD RUNOFF (DSR) PROGRAM

The DSR is a document formatter that is available on all major DEC systems.

1.2.1 DSR Commands and Flags

Specific format control is accomplished by inserting DSR commands before the line(s) of text to be formatted. Additional enhancements (e.g., underlining) are then specified by imbedding flags (special characters) within the text.

DSR Command Recognition

Command recognition by DSR is achieved by the insertion of a period (CONTROL flag) before each command.

If a single command is entered on a line, the period appears in physical column one. The command is then terminated by an end-of-line code (e.g., RETURN) or a special terminator character (e.g., semi-colon), and the normal text is entered.

If more than one command is entered on a line each subsequent command is separated from the previous command by its CONTROL flag (.). This continues until the command string is terminated, as described, at which point normal text is entered.

Thus, the period at the beginning of a line enables the recognition of commands and the terminator disables recognition, allowing the insertion of normal text.

1.2.2 DSR Default Modes

When an input file is processed by the DSR, certain default actions are performed that do not depend on command or flag entries for their execution. These actions are closely analagous to actions performed during the preparation of a manually typed document.

For instance, regardless of the complexity of a document, the first considerations of the typist are: paper size, mechanical margin settings, line spacing, possible tab settings, and the shape of the subject matter as it must appear in the final draft.

Based on these considerations, the DSR default modes provide the following:

- A standard typewriter paper (or page) size of 8 1/2 X 11 inches, accommodated by defining a width of 60 character columns and a length of 58 lines of text per page.

- Sequential page numbering for every page but the first. This provides for the possibility of an unnumbered single-page document or an unnumbered title page for a multi-page document.

- A predefined left margin of 0 (first character position) and right margin of 60 (last character position) which provides for a basic line width of 60 characters.

- A line spacing equivalent to the single-space setting on a typewriter.

- Automatic tab settings for every eighth print position, starting with the ninth print column (9, 17, 25, etc.).

- Automatic Filling: where successive words from the source text are added to a line in the object file until the addition of another complete word would exceed the right margin.

- Automatic Justifying: where spacing between words is adjusted to move the last word in the line over to the right margin, exactly. The overall effect is an even (as opposed to a ragged) right margin.

To familiarize the reader with using the DSR formatter, the following subsections provide step-by-step procedures for processing and printing two sample documents, including descriptions of the processes. The reader is encouraged to follow the directions and create these sample files.

The following assumes that the reader has a working knowledge of TOPS-10 and SOS. Users of other systems and editors must make adjustments for the differences.

1.2.3 DSR Formatting Example 1

To acquaint the reader with DSR default processing, the following example depicts the entry of text into the input file without the insertion of DSR commands or flags.

Use steps 1 and 2 to create a source file having the file specification DEM01.RNO; input the example, and then use steps 3 and 4 to create and print the output file.

Creating the Source File

1. Summon the appropriate text editor (refer to system usage manual) and create or access the input file.

2. Input the text.

Input text for Example 1:

```
00100  Oh, East is East, and West is West, and never the
00200  twain shall meet,
00300  Till Earth and Sky stand presently at God's great
00400  Judgement Seat;
00500  But there is neither East nor West. Border, nor
00600  Breed, nor Birth,
00700  When two strong men stand face to face, though they
00800  come from the ends of the earth!
00900                  The Ballad of East and West
```

Creating the Object File

3. Create a draft of the document as follows:

```
.RUN RUNOFF<CR>
*DEMO1<CR>
No errors detected by version 1(71) of RUNOFF
1 page written to DSK:DEMO1.MEM
```

The input file (DEMO1.RNO) is now processed and an output file created, to which DSR affixes an output file extension (DEMO1.MEM).

If any error is detected during processing, the type of error and the input and output location of the error will be printed. If an error occurs return to Step 1 for corrective action, if not proceed to Step 4.

4. Print the document as follows:

```
.PRINT DEMO1.MEM<CR>
```

The Example 1 output should be:

Oh, East is East, and West is West, and never the twain shall meet, Till Earth and Sky stand presently at God's great Judgement Seat; But there is neither East not West, Border, nor Breed, nor Birth, When two strong men stand face to face, though they come from the ends of the earth! The Ballad of East and West

In the print-out of the final version, notice the following:

The format used for the input file has been ignored by DSR in producing the output file. Without specific DSR commands only applicable default values have been used to shape the text. Thus, in the output file, default margins have been set, single-spacing has been used, and the text has been filled and justified (refer to DSR Default Modes).

1.2.4 DSR Formatting Example 2

To acquaint the reader with DSR command and flag entry processing, the following example depicts the entry of text into the input file along with certain DSR commands and flags.

Repeat steps 1 and 2 to create a source file having the file specification DEMO2.RNO; input the example, and then use steps 3 and 4 to create and print the output file.

Input text for Example 2:

```
00100  .B;Oh, East is East, and West is West, and never the
00200  .B.I2;twain shall meet,
00300  .B;Till Earth and Sky stand presently at God's great
00400  .B.I2;Judgement Seat;
00500  .B;But there is neither East nor West, Border, nor
00600  .B.I2;Breed, nor Birth,
00700  .B;When two strong men stand face to face, though they
00800  .B.I2;come from the ends of the earth!
00900  .B2.L16;^&The Ballad of East and West\&
```

The Example 2 output should be:

Oh, East is East, and West is West, and never the
twain shall meet,

Till Earth and Sky stand presently at God's great
Judgement Seat;

But there is neither East nor West, Border, nor
Breed, nor Birth,

When two strong men stand face to face, though they
come from the ends of the earth!

The Ballad of East and West

The following provides a brief description of the DSR commands, and flags, previously used to affect the final formatting of the output file:

```
.                   ;Command CONTROL flag (.)

.B                  ;leave one Blank (single-spaced)
                    ;line

.I2                 ;Indent the line by two columns

.L16                ;Set spacing for title of poem

^                   ;Lock UNDERLINE (for line) flag

&                   ;UNDERLINE flag

\                   ;Unlock UNDERLINE (for line) flag
```

In the print-out of the final version, notice the following:

- A CONTROL flag (.) has introduced various DSR commands (i.e., .B, .I2, .L16), optionally entered in upper case.

- The .BLANK (.B) command has caused a blank line to occur before each line of text (replacing default single-spacing).

- The .INDENT 2 (.I2) command has caused each alternate line of text to be indented two spaces from the left margin.

- The .LEFT 16 (.L16) command has caused 16 spaces to precede the title.

- DSR flag characters have been used to first turn on underlining (^&) and then turn it off (\&).

CAUTION

Once a document has been printed, the .MEM file may be destroyed (since it can be re-created from the .RNO file). However, if the .RNO file is destroyed further editing will not be possible.

1.3 LEARNING TO USE THE DSR LANGUAGE

Since DSR can recognize over 100 commands and flags, an attempt to memorize and sift through such numbers, to select the proper control, can be chaotic and time consuming.

To somewhat alleviate this problem, command descriptions are arranged (Chapter 3) in general-usage groups (e.g., page, title, section formatting, etc.) and described in a sequence as closely related to the logical preparation of a document as possible.

CHAPTER 2

DSR RULES AND CONVENTIONS

2.1 DSR COMMANDS

The following material describes the general formatting of the DSR commands, including basic rules for entry and conventional usage. As stated, this material emphasizes only the basic requirements. For more precise syntactical descriptions, concerning required and optional formatting, refer to the command listing in APPENDIX C.

A DSR command consists of the following parts:

1. A required period (.) in column one, used by the DSR to identify the following word as a command.

2. A required keyword (e.g.,BLANK) or keywords (e.g., LEFT MARGIN), directly following the period.

3. Parameters, as required, to provide additional information.

4. A terminator character (e.g., semi-colon) to specify the end of a command.

2.1.1 Periods

The period (.) is an integral part of the structure of DSR commands. When a period is used to provide for command recognition, it is formally defined as a CONTROL flag.

2.1.2 Keywords

Keywords are used to direct the DSR program to perform a specific action. Each keyword is entirely composed of letters (A-Z), and a single command may consist of multiple keywords. However, only the first keyword of a command is preceded by a DSR CONTROL flag (.), while any additional keywords are separated by spaces (i.e., depressions of the SPACE bar or TAB key).

Thus, a three-keyword command appears as:

```
.NO CONTROL CHARACTERS
```

When more than one command is entered on the same line the first CONTROL flag (.) is inserted in column one, directly followed by the first keyword. Periods are then used to both terminate the current command and introduce the first keyword of the following command.

```
.NO FILL.LEFT MARGIN 0.RIGHT MARGIN 40
```

Note that at least one space (or tab) must be used to separate keywords; additional spaces (and/or tabs) are optional.

2.1.2.1 Keyword Entry Forms

The letters of a keyword may be entered in any case (e.g., .HEADER, .HEAder, .header). However, the keywords themselves must be entered in their complete word form, in an acceptably truncated (shortened) form, or in a legally abbreviated form (e.g., .hd).

Legal abbreviations for the previous examples follow:

```
.NCC

.NF.lm0.Rm40
```

The truncation of a keyword (in which the word is reduced from right to left, letter by letter) is acceptable when the resultant form can be interpreted as a shortened word and the form does not conflict with another command.

For example, the following truncations are acceptable:

```
.NO CONTROL CHARACTER

.NOCON CHA

.NOCO CH

.N C C
```

However, the following truncations are illegal because: in the first case, the word "CHARACTER" cannot be interpreted without a preceding space, and in the second case, the word "NO" cannot be interpreted:

```
.NOCOCH

.NC C
```

Since the number of possible truncations is great, truncation should be avoided or, at least, used with discretion.

NOTE

1. All the abbreviations used in this manual are legal and will be accepted by all versions of DSR.

2. Truncated forms currently acceptable to DSR may not be acceptable to later versions of DSR.

2.1.2.2 Recognition and Scanning

When the DSR program recognizes a CONTROL flag (.), it immediately scans each keyword, directly following the period, to identify the command.

If any parameters are required, the recognition of the command will only be possible if the character directly following the last letter of the final keyword is not another letter. The character must, therefore, be non-alphabetic (e.g., a leading digit).

2.1.2.3 Keyword-parameter Separation

The keyword-parameter separation rule is as follows: If the leading character of the first parameter is not a letter (A-Z) no separation is required; but if the leading character is a letter, the letter must be separated from the final keyword by at least one space or tab.

In the following examples, the keywords are identifiable because each associated parameter can be properly differentiated from its keyword: first by a digit, second by a special character, and third by a space-separator.

```
.TITLE400MODE

.TITLE$COPY COMMAND

.TITLE RUNOFF
```

In the next example the parameter cannot be differentiated:

```
.TITLERUNOFF
```

2.1.3 Parameters

Depending on the type of command, the parameters may be any (or in some cases a combination) of the following:

- o number-parameter (n)
- o count-parameter (c)
- o text-parameter (textl or text2)
- o character-parameter (k)
- o quoted-string-parameter (q)
- o display-descriptor-parameter (y)
- o name-parameter (name)
- o draft-flag-parameter (flags)

2.1.3.1 Parameter-parameter Separation

The parameter-parameter separation rule is as follows: If more than one parameter is required, a space- or comma-separator may be inserted between parameters. However, if differentiation between parameters is not possible (i.e., due to digit to digit or letter to letter adjacency) then a separator must be inserted.

Space- and comma-separators are respectively defined as:

1. at least one depression of the SPACE bar or TAB key.
2. a single comma, alone or within any number of spaces and/or tabs.

In the following examples, the first and second parameters (i.e., 1 and 1ABC) are properly differentiated by the use of a space and comma separator.

```
.HEADER LEVEL1 1ABC

.HEADER LEVEL1,1ABC
```

In the next example the parameters cannot be differentiated:

```
.HEADER LEVEL11ABC
```

Note in the last example that DSR cannot discern a level 1, with a 1ABC title, from a level 11 with an ABC title.

Null Parameters

If the user, while entering a sequence of parameters, desires to evoke a default value, a comma-separator must be used to indicate the insertion of a null parameter (i.e., a non-specific value), to which DSR will assign an appropriate default value.

For example, the .TAB STOPS command can accept several number-parameters. However, where a combination of number and null parameters are used, the command will take specified values and evoke default values on a positional basis (e.g., 8, 16, 24, 32, etc.) whenever the TAB key is depressed. In the following example, the command invokes tab stop 10 (specified) the first time the TAB Key is depressed, and (due to the extra comma) defaults to tab stop 16 for the null parameter the second time the key is depressed.

Format: .TAB STOPS n1,n2...n32

Example: .TS10,,20,26,30

The same entry may be made as follows:

```
.TS10,,20 26 30
```

In the next example, the .DISPLAY ELEMENTS command invokes default values for the first ("q") and second (y) parameters, via commas, while using a specific value (x) for the third parameter ("q").

Format: .DISPLAY ELEMENTS "q",y,"q"

Example: .DLE ,,"x"

2.1.3.2 Number-Parameter (n)

Number-parameters consist of signed (+/-) or unsigned decimal (0-9) numbers, such as:

```
.SKIP 2

.INDENT +4

.INDENT -12
```

Unsigned Numbers

Excluding leading zeroes, unsigned numbers may be up to four decimal digits in length (0 thru 9999).

These numbers are used to provide for the direct assignment of a value (i.e., they cannot add or subtract a value).

The following commands use unsigned numbers:

.AUTOSUBTITLE	.NUMBER LIST
.CENTER	.NUMBER PAGE
.FIGURE	.NUMBER SUBPAGE
.FIGURE DEFERRED	.PAGE SIZE
.FOOTNOTE	.PARAGRAPH
.HEADER LEVEL	.REPEAT
.INDENT	.RIGHT
.LEFT	.RIGHT MARGIN
.LEFT MARGIN	.SKIP
.LIST	.SPACING
.NUMBER APPENDIX	.STANDARD
.NUMBER CHAPTER	.TAB STOPS
.NUMBER LEVEL	.TEST PAGE

Signed Numbers

Excluding leading zeroes, signed numbers (+/-) may be up to four decimal digits in length (+/- {1 thru 9999}).

These numbers are generally used to add or subtract a desired value to or from a value already assigned by a previous command.

When a signed number is entered, it will only be valid if the first digit of the number directly follows the sign. Thus, +22 and -999 are valid, while + 22 and - 999 are not.

The following example depicts the addition and then the subtraction of left margin values:

```
.LEFT MARGIN10    !set left margin to 10.
      .
      .
.LEFT MARGIN+6    !set left margin to 16.
      .
      .
.LEFT MARGIN-6    !set left margin back to 10.
```

The following commands may use signed number-parameters.

.AUTOSUBTITLE	.NUMBER LIST
.CENTER	.NUMBER PAGE
.HEADER LEVEL	.NUMBER SUBPAGE
.INDENT	.PAGE SIZE
.LEFT	.PARAGRAPH
.LEFT MARGIN	.RIGHT
.NUMBER APPENDIX	.RIGHT MARGIN
.NUMBER CHAPTER	.SKIP*
.NUMBER LEVEL	.STANDARD
	.TAB STOPS

*When a signed number is used with a .SKIP command only a direct assignment can be made (i.e., no change in current value can occur).

2.1.3.3 Count-parameter (c)

The count-parameter can be either a letter string or a number string. The parameter is used to start a new numeric count sequence by forcing the counter to the value specified.

If letters are used they are equated with associated numeric values (i.e., a through z = 1 through 26). For values greater than 26 the letters are repeated, as follows:

aa - az = 27 - 52

ba - bz = 53 - 78

ca - cz = 79 - 104

(etc.)

Moreover, signed numeric values may be used to adjust the current value by addition or subtraction.

Understand that the use of the count-parameter alters numeric values, but does not cause letters to appear in the output; the latter can only occur as the result of a .DISPLAY command (refer to 2.1.3.7).

The following commands use the count-parameter:

```
.NUMBER APPENDIX
.NUMBER CHAPTER
.NUMBER PAGE
.NUMBER SUBPAGE
```

2.1.3.4 Text-Parameter (textl or text2)

There are two types of text-parameters (textl and text2). Textl may be composed of any characters except end-of-line (EL), while text2 may be composed of any characters except end-of-line and a semi-colon (;).

However, the acceptance of a semi-colon in text2 (or any special character except an end-of-line code) can be forced by preceding the character with an ACCEPT flag (_). As with all flags, the flag character will not appear in the output file. Consider the following examples:

Format: .CHAPTER textl

Example:

```
.CH DSR COMMANDS          !define new chapter and title
```

Format: .HEADER LEVEL n text2

Examples:

```
.HL1 1ABC                 !defines n = 1 and text = 1ABC
.HL3 _$COPY               !flag(_) allows text2 = $COPY
```

The following commands use textl:

```
.APPENDIX
.CENTER
.CHAPTER
.DO INDEX
.ENTRY
.NOTE
.RIGHT
.SUBINDEX
.SUBTITLE
.TITLE
```

The following commands use text2:

.COMMENT
.HEADER LEVEL
.INDEX

2.1.3.5 Character-Parameter (k)

The character-parameter is only used with a .FLAGS command to replace an existent flag character with another.

Only a single character is allowed, and the character must be separated from the last keyword (flag name) as defined by the keyword separation rule:

Format: .FLAGS name k

Example: .FL ACCEPT %

Certain flags may also be used with a specified replacement character for either enhancement (i.e., UPPERCASE or LOWERCASE) or validation (ACCEPT) purposes.

For example, the first of the following two commands uses an UPPERCASE flag (^) to provide a replacement character that is identical to that specified by the second command.

.FLAGS UNDERLINE ^a

.FLAGS UNDERLINE A

In this example, the ACCEPT flag is used to force the recognition of a control character that would otherwise be rejected by DSR.

.FLAGS OVERSTRIKE _(BS)

2.1.3.6 Quoted-String-Parameter (q)

The quoted-string-parameter consists of up to 150 characters, enclosed in either single (') or double (") quotes.

A quoted-string is variously used to specify the type of bullet character for a list, a graphic display for replication, or a filename specification.

In the following example, the parameter is used with the .LIST command to define both the bulleting feature and the type of bullet that will appear (i.e., a lower case o).

Format: .LIST n,"q"

Example: .LS 2,"o"

Notice, in the example, that the number-parameter is optionally separated from the keyword by a space while the quoted-string is optionally separated from the number-parameter by a comma.

In the next example, the parameter is used with a .REQUIRE command to specify an external file that is required as input to the user's file:

Example: .REQUIRE "ABC.RNO"

The following commands use the quoted-string-parameter:

.LIST
.REPEAT
.REQUIRE
.DISPLAY ELEMENTS

2.1.3.7 Display-Descriptor-Parameter (y)

A display-descriptor-parameter is a one- or two-character descriptor code (see Table 2-1). The codes are used with the .DISPLAY commands to change the default numbering format (e.g., decimal), evoked by DSR, to a format specified by the user.

Understand that the parameter only affects a change of format not a change of value.

Table 2-1

DISPLAY DESCRIPTOR CODES

DESCRIPTOR (y)	PURPOSE
D	Use decimal numbering.
O	Use octal numbering.
H	Use hexidecimal numbering.
RU	Use upper case roman numerals.

RL	Use lower case roman numerals.
RM	Use mixed case roman numerals. (only first numeral is upper case)
LU	Use upper case letters.
LL	Use lower case letters.
LM	Use mixed case letters. (only first letter is upper case)

Note that the case of roman numerals and letters is specified by the second descriptor code character. Further, when using the parameter, the codes may be entered in any case (e.g., RU, rU, Ru, ru).

The following example shows the parameter used to specify upper case roman numerals for chapter numbering:

Format: .DISPLAY CHAPTER y

Example: .DCH RU

The following commands use the display-descriptor-parameter:

.DISPLAY APPENDIX
.DISPLAY CHAPTER
.DISPLAY ELEMENTS
.DISPLAY LEVELS
.DISPLAY NUMBER
.DISPLAY SUBPAGE

2.1.3.8 Name-Parameter (name)

A name-parameter is a sequence of up to 15 alphanumerics (combination of letters and digits), the first character of which must be a letter. Therefore, the parameter must be separated from the keyword by a space.

Format: .ELSE name

Example: .ELSE ABC1

The following commands use the name-parameter:

.ELSE
.ENDIF
.IF
.IFNOT
.VARIABLE

2.1.3.9 Draft-Flag-Parameter (Flags)

The draft-flag-parameter is exclusively used by the .VARIABLE command, and always follows its name-parameter.

A draft flag may be any alphanumeric or special character except a space, a comma (,), or an end-of-line (EL) code.

If differentiation is a problem, the first (or only) flag must be separated from the name-parameter by a space. If a second flag is required it may be optionally entered back-to-back with the first or separated by a space or a comma (,).

Format:	.VARIABLE name flags
Examples:	.VR BARS 12
	.VR BARS 1 2
	.VR BARS T,F

2.1.4 Command Terminators

There are four command termination possibilities: any one of three characters (i.e., period, semi-colon, exclamation mark) and/or an end-of-line code (EL).

Terminator Purpose

Terminators provide the user with a time and space saving convenience, while accommodating DSR's need to distinguish between individual commands and commands and text. To understand this, recall that the basis of DSR command identification is the recognition of a CONTROL flag (.) in column one. Therefore, in order to provide the convenience of including commands or commands and subject matter on the same line, the insertion of an appropriate terminator serves not only to end a command but to logically simulate column one.

Terminator Scan

When DSR has scanned and processed the parameter(s) associated with a command, the program searches for a legal terminator, ignoring any intervening spaces or tabs on the line.

The search is valid if:

1. an end-of-line (EL) code is found, indicating there are no more characters on the line, or

2. a period (.), semi-colon (;), or exclamation mark (!) is found, indicating the nature of the next entry.

Restrictions and Syntax

The insertion of a specific terminator, therefore, depends on the requirements of the command to be terminated, the nature of the next entry and/or the manner in which it will be entered.

For example, the first character entry following a .REQUIRE command parameter must occur on the next line; therefore, the command can only be terminated by an end-of-line (EL) code:

```
.REQUIRE "ABC.RNO"{EL}
```

For these reasons, the following examples depict the manner in which command termination possibilities are denoted throughout this manual. Note that choices are shown enclosed in braces ({}), with each choice being separated by a vertical bar (|) which symbolizes the word "or".

```
.END SUBPAGE{.|EL|;|!}

.FOOTNOTE n{EL|;}

.TITLE textl{EL}
```

2.1.4.1 End of Line (EL)

An End-of-Line (EL) code acts as a terminator for a command (or text) and begins a new line at physical column one.

The term (end-of-line) identifies all codes which introduce a new line (e.g., carriage return, line feed, etc.).

```
.B.NF.LM0.RM40{EL}
```

2.1.4.2 Period (.)

The period acts as a terminator for the current command and introduces the next command on the same line.

This character causes DSR to consider its placement as being at column one regardless of where on the line the character actually occurs, for example:

```
.B.NF  .LM0        .RM40{EL}
```

Notice that each period represents column one regardless of the occurrence of two spaces and a tab (8 spaces) within the string of commands.

2.1.4.3 Semi-Colon (;)

This character causes DSR to consider the next character as being in column one; regardless of where on the line the character actually appears.

This satisfies the program in regard to:

1. The placement of subject matter in column one, regardless of the fact that the text is entered on the same line.

2. The placement of a string of certain commands (e.g., .INDEX) on the same line, for which only the semi-colon or an end-of-line code is a legal terminator.

Two examples follow: The first shows the semi-colon terminating a command and introducing subject matter, the second shows .INDEX commands used on the same line:

```
.SKIP;text follows              !first text character is in
                                !logical column one.

.INDEX text2;.INDEX text2       !CONTROL flags are in
                                !physical and logical column one.
```

2.1.4.4 Exclamation Mark (!)

When this character is used as a terminator for a command it also acts as a COMMENT flag, that is used to introduce user commentary on the same line (visible in the input but not the output file).

The following examples depict the use of the exclamation mark as both a terminator/COMMENT flag and a replacement for a .COMMENT command.

```
.SKIP!commentary          !"c" is in logical column one
                          !the flag is not.

.!commentary              !"c" is in logical column one.
```

WARNING

The exclamation mark (!) must not be used in physical or logical column one within a footnote (for details refer to APPENDIX A).

2.2 INTRODUCTION TO DSR FLAGS

A flag is a uniquely named (e.g., UNDERLINE) character (e.g., an ampersand) that is used to perform a specific operation (e.g., underlining).

The specified operation is invoked when the character is entered in the text and recognized as a flag by DSR.

When a flag character is imbedded in the text its operation can only be performed if:

1. The character is recognized as a flag by DSR, and

2. the operation is enabled, either by default or via an associated DSR command.

When a character is recognized by DSR as a flag it will not appear in the output, even if its operation is disabled. However, if a character is not recognized it will appear as normal text and its operation, even if enabled, will not be performed.

Certain of the flag characters are initially recognized by default (refer to Table 2-2) and, with the exception of the SUBINDEX flag, are operable without the use of a command.

The remaining flags provide advanced usage and require that character recognition be enabled by command. However, once recognition is achieved the flag is operable.

In the following material, the basic rules governing the use of all flags are described, followed by brief descriptions of the default flags with examples of their use.

For a complete analysis of flag operations refer to Chapter 4 (DSR FLAGS and COMMAND CONTROL).

2.2.1 Flag Recognition and Control

With the exception of the CONTROL and COMMENT flags, a .NO FLAGS ALL command can be used to collectively disable the recognition of all of the flags, while a .FLAGS ALL command can be used to re-enable them.

In addition, each flag can be individually disabled or re-enabled by using its name as the parameter (instead of ALL) for the .FLAGS command.

The following rules apply to flag recognition control:

- If the recognition of a flag is disabled (either collectively or individually) its character will be appear as normal text.

- If recognition is enabled (both collectively and individually) the execution of the flag's function will depend on whether or not the flag's usage is under the control of a formal command.

For example, when an UNDERLINE flag (&) is inserted before a character it will cause the character to be underlined in the output. However, if a .NO FLAGS UNDERLINE command is inserted prior to the use of the flag, it will disable the recognition of the flag's character (&), prevent underlining, and cause the character to appear in the output as normal text. Subsequently, if a .FLAGS UNDERLINE command is entered, recognition of the flag is re-enabled and underlining will again occur.

For details of all flag command operations refer to Chapter 4.

Redefining a Flag Character

For special purposes, a flag character may be redefined by character replacement.

For example, the following shows the redefining of the SPACE flag character (by default a number sign) to a dollar sign via the .FLAGS command.

```
.FLAGS SPACE $
```

Since the redefinition of a flag will affect any existent paired condition of the character, and present difficulties to subsequent users of the source file, such practice is not recommended.

Character Replacement Rule

However, if character replacement is necessary, the user must be aware of the following:

To recognize a flag, DSR scans all of the default characters in the sequence shown in Table 2-2. Therefore, if the level of recognition of a flag character precedes its use as a replacement character, its initial use must be disabled in order for recognition as a replacement to occur.

For example, if the UPPERCASE flag character, by default a circumflex (^), is to be replaced by an underscore (_), which is the default character for an ACCEPT flag, ACCEPT must first be disabled (.NO FLAGS ACCEPT) for the underscore to be accepted as an UPPERCASE flag replacement (refer to Chapter 4.2 FLAG CONTROL COMMANDS for example).

Finally, all flag descriptions and examples used in this manual reference the DSR default flag characters.

2.2.2 The Default Flags

In Table 2-2, the default flags and normally associated characters are listed, along with the general function of each.

Table 2-2

DEFAULT FLAG DESCRIPTIONS

FLAG NAME	CHARACTER	PURPOSE
CONTROL	.	introduce DSR command
COMMENT	!	begin a comment
ACCEPT	_	take next character as normal text
UPPERCASE	^	upper case next character
LOWERCASE	\	lower case next character
UNDERLINE	&	underline next character
SPACE	#	force unexpandable space
SUBINDEX	>	insert next word in index

With the exception of the SUBINDEX flag, all of the functions listed in Table 2-2 are operable without the use of DSR commands.

In regard to the exception, the SUBINDEX character (>) is inoperable as a SUBINDEX flag unless it is used within an .INDEX or .ENTRY command parameter.

The following provides a brief description of each default flag.

2.2.2.1 The CONTROL Flag (.)

The single occurrence of this character in physical or logical column one allows the character to be recognized as a flag and the keyword which follows to be recognized as a DSR command.

The CONTROL flag may be paired in two ways:

- With a COMMENT flag (.!), to introduce user commentary at the start of a line.
- Following its occurrence in column one, the flag may be paired with itself (..) to force the flag character to be taken as normal text.
- Following an ACCEPT flag in column one (_.), which forces the flag character to be taken as normal text.

2.2.2.2 The COMMENT Flag (!)

The COMMENT Flag can be used in place of the COMMENT keyword to allow the insertion of user comments in and for the .RNO file (i.e., the commentary will not appear in the output file).

When the flag is inserted in a multi-command string it becomes both the terminator of the previous command and the initiator of the commentary, for example:

```
.LMO.RM60!place comment here
```

The flag may be paired as follows:

- Following a CONTROL flag, to introduce a comment at the start of a line:

```
.!place comment here
```

- Following an ACCEPT flag (_!), which forces the character to be taken as normal text.

For additional examples of the use of the flag, refer to section 2.1.4.4 Command Terminators (exclamation mark).

2.2.2.3 The ACCEPT Flag (_)

This flag causes any character that directly follows it (including flags and control characters) to be accepted as normal text.

In addition, for underlining purposes, the flag can be used to ACCEPT a space, thereby allowing the space to be accepted as a normal character.

When the flag is paired with itself (__) the flag character will be taken as normal text.

2.2.2.4 The UPPERCASE Flag (^)

As a single character flag, UPPERCASE serves the same purpose as a typewriter shift key, and will capitalize any single letter that directly follows it. The flag has no effect if the following character is not a letter.

The UPPERCASE flag may be paired in three ways:

- With an UNDERLINE flag (^&), to initiate the underlining of the text it precedes .
- With itself (^^), to affect a case-lock (i.e., no case change).
- Following an ACCEPT flag (_^), which forces the flag character to be taken as normal text.

In regard to case-lock: setting a case-lock on a fixed-case terminal will invoke an upper case mode, while setting a case-lock on the variable-case terminal will invoke a lower case mode. This occurs because the normal mode of operation for each type of terminal will be imposed via a no-case-change (refer to section 4.1.7 Fixed-Case Terminals).

2.2.2.5 The LOWERCASE Flag (\)

The LOWERCASE Flag causes the letter that directly follows it to appear in lower case. The flag has no effect if the following character is not a letter.

The LOWERCASE flag may be paired in three ways:

- With the UNDERLINE Flag (\&), to terminate the underlining of text.
- With itself (\\), to affect a lower-case-lock (i.e., force lower case) on all subsequent letters (refer to 4.1.7 Fixed-Case Terminals).
- Following an ACCEPT flag (_\), which forces the flag character to be taken as normal text.

2.2.2.6 The UNDERLINE Flag (&)

The single character occurrence of this flag allows the next character to be underlined, for example:

entering:	Use Type &C
produces:	Use Type C

The flag may also be used to underline a single real space between characters:

entering:	Underline& single& spaces.
produces:	Underline_single_spaces.

Note that when the flag is used in this manner it cannot be preceded by a real space or the effect will be cancelled.

The UNDERLINE flag may be paired as follows:

- Preceding an ACCEPT flag that forces acceptance of a space (&_), to generate an underlined space:

entering:	Underline&_ this&_ &_ spacing.
produces:	Underline_this__spacing.

- Preceding a SPACE flag (&#), to underline unexpandable spaces:

  entering: Underline&#this&#&#&#spacing.

  produces: Underline_this___spacing.

- Following an UPPERCASE flag (^&), to lock underlining, and following s LOWERCASE flag (\&) to unlock underlining:

  entering: ^&Underline these words\&

  produces: Underline these words

- Following an ACCEPT flag (_&), which forces the flag character to be taken as normal text.

Finally, paired and single character usage can be combined as follows:

entering: ^&Character&#and&#Word\&

produces: Character and Word

2.2.2.7 The SPACE Flag (#)

This flag forces an unexpandable space (not affected by justification) to appear in the output every time the character is inserted in the input.

If the spacing between two consecutive words consists of SPACE flags, and no real spaces or end-of-line code interrupts them, justification of the line will not affect the spacing between the words, for example:

entering: These#spaces##are###inviolate.

produces: These spaces are inviolate.

The flag may be paired in two ways:

- Following an UNDERLINE flag (&#), to affect the underlining of the space (refer to UNDERLINE Flag).

- Following an ACCEPT flag (_#), which forces the flag character to be taken as normal text.

2.2.2.8 The SUBINDEX Flag (>)

Although this flag is initially enabled by default, it is only operable when it is used to delimit parameters from an INDEX, SUBINDEX, or ENTRY command, for an index.

When used in this manner, the SUBINDEX flag allows all the parameter characters that directly follow it to be included in the index until the end of a parameter is indicated by the appearance of:

- a semi-colon (;).

- an end-of-line (EL) code.

- another SUBINDEX flag (indicating another parameter for the index).

For details refer to Chapter 6, CREATING AN INDEX.

When paired with the ACCEPT flag (_>), the flag character is taken as normal text.

2.3 DSR SPACING CONVENTIONS

The effects of spacing with DSR may be summarized as follows:

- Spaces inserted by the SPACE Bar or TAB key for the differentiation of keywords and/or parameters are otherwise ignored by DSR.

- Spaces inserted in normal text, by SPACE Bar or TAB Key, are (if justification is enabled) both expandable and disposable, unless encompassed by a .LITERAL command structure.

- Spaces inserted in normal text via a SPACE flag (#) cannot be affected by DSR and are, therefore, inviolate.

CHAPTER 3

THE BASIC DSR COMMANDS

3.1 PAGE FORMATTING COMMANDS

The DSR program provides a Paging Mode of operation by default. With this operation, all documents (non-sectioned or sectioned) are automatically split into sequentially numbered pages.

The user may enter page formatting commands to specify the following:

- The size of the text area in relation to the physical size of the paper.
- The appearance and enhancement of top-of-page running head information (included as part of the text area).
- The disabling of an element of the Paging Mode (i.e., numbering) or the mode itself.
- The optional initiation and termination of subpaging within the Paging Mode.

3.1.1 Page Size and Header Enhancement

Page Sizing

Page or paper sizing provides for the uniform positioning of text on each page of the printed document.

Sizing is accomplished by specifying the maximum size of the text in columns of width and in lines of length. Since the size of the allowed text may range to a maximum width of 150 columns and a maximum page length of 9999 lines, any standard printer paper size (e.g., 14 by 11 inches) may be accommodated.

Once the size of the text is established, the basic width and length may be reduced or expanded, as required, to shape the document.

Since line printers use standardized horizontal paper positioning and software controlled vertical positioning (page breaks), page size default values are available (60 columns by 58 lines) that are predicated on the general use of a standard typewriter paper size of 8 1/2 by 11 inches (See Figure 3-1A).

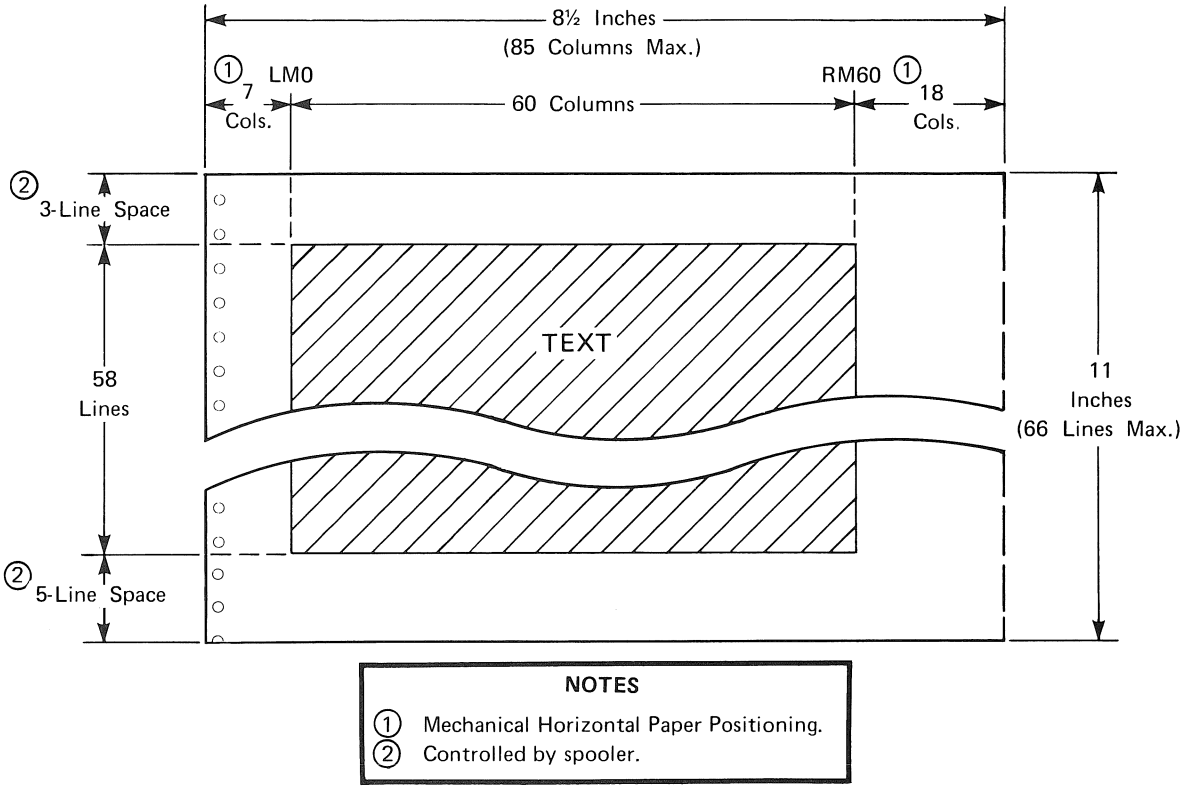


Figure 3-1A Line Printer Form Layout (58 Lines, 60 Columns)

As shown in Figure 3-1B, the default page length (58 lines), in conjunction with the vertical positioning of the device, automatically allows for appropriate spacing at the top and bottom of the page. However, the actual number of printed lines appearing on a page depends on the number of blank lines introduced in the text. This statement is not as obvious as it seems, since many blank line insertions are not imposed by user but by the DSR to satisfy page, chapter, and running head requests (See Figure 3-2).

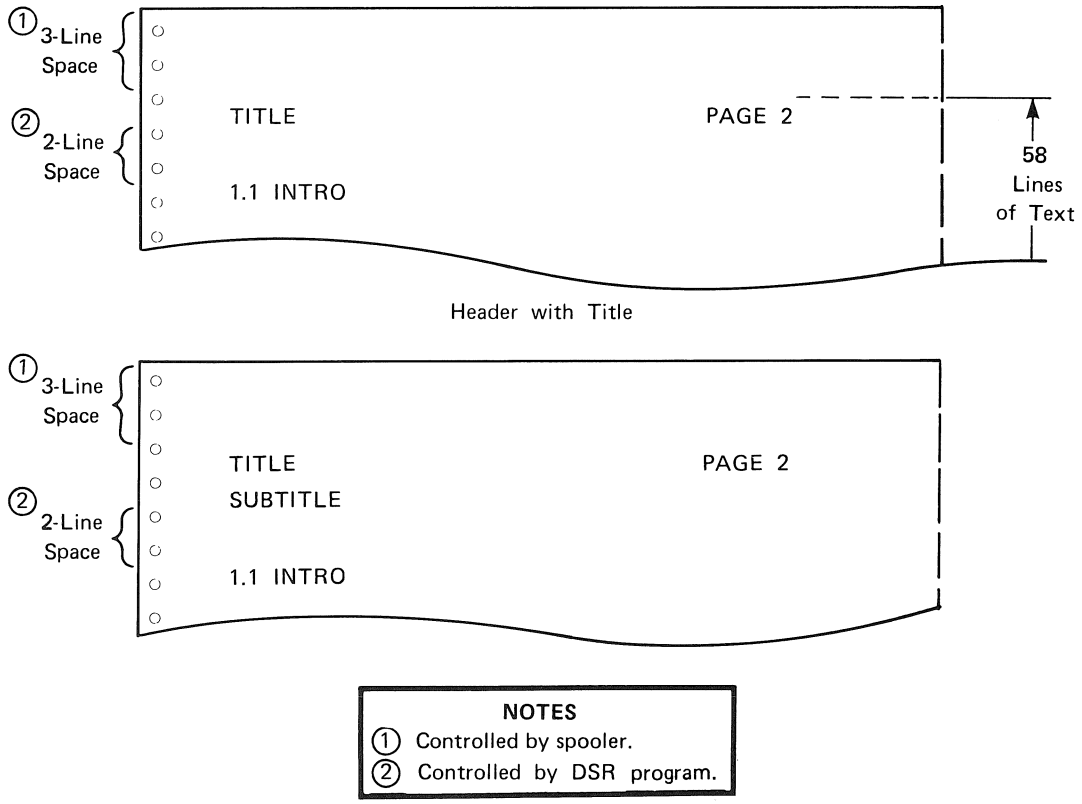


Figure 3-1B Header with Title and Subtitle

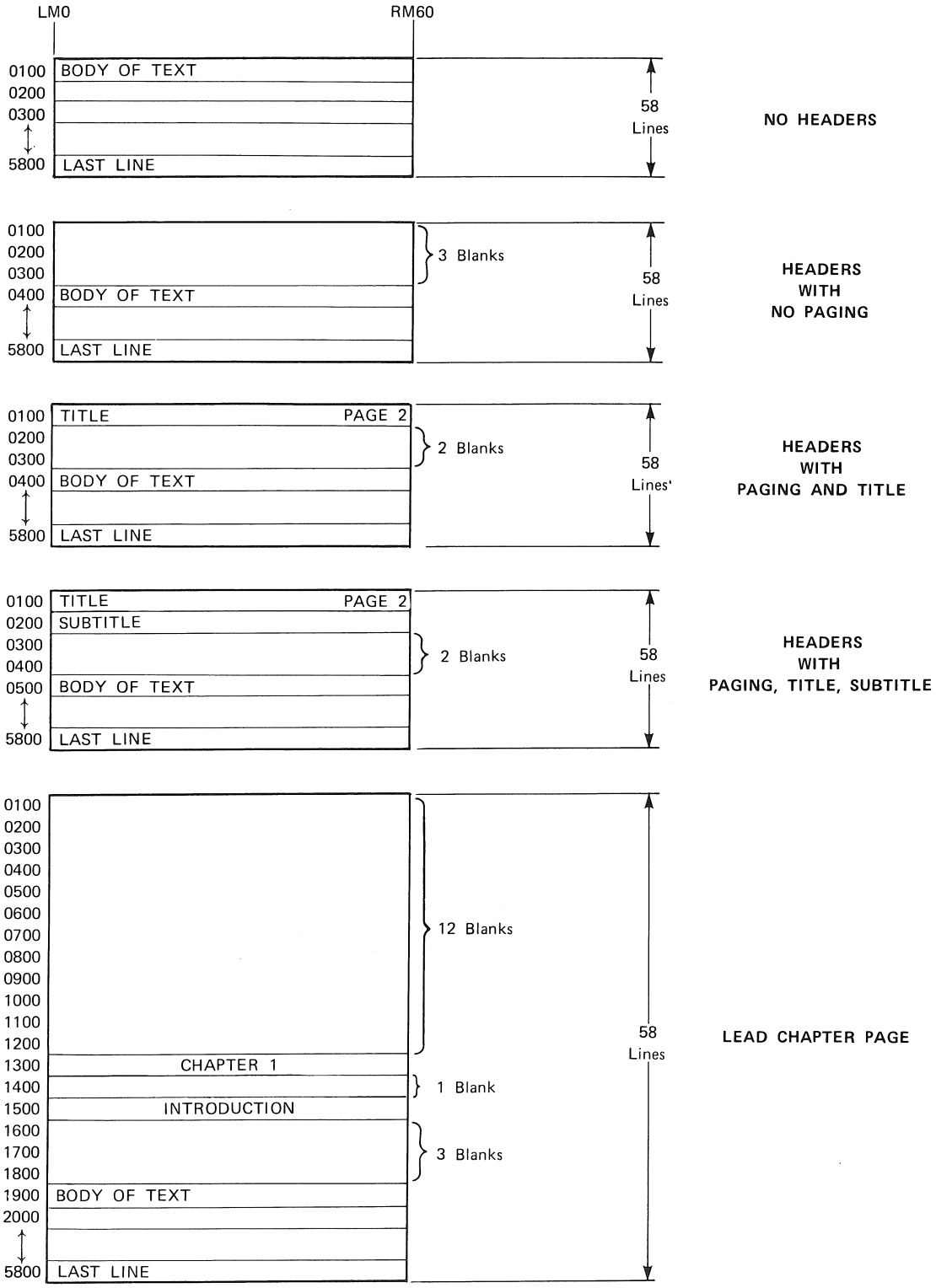
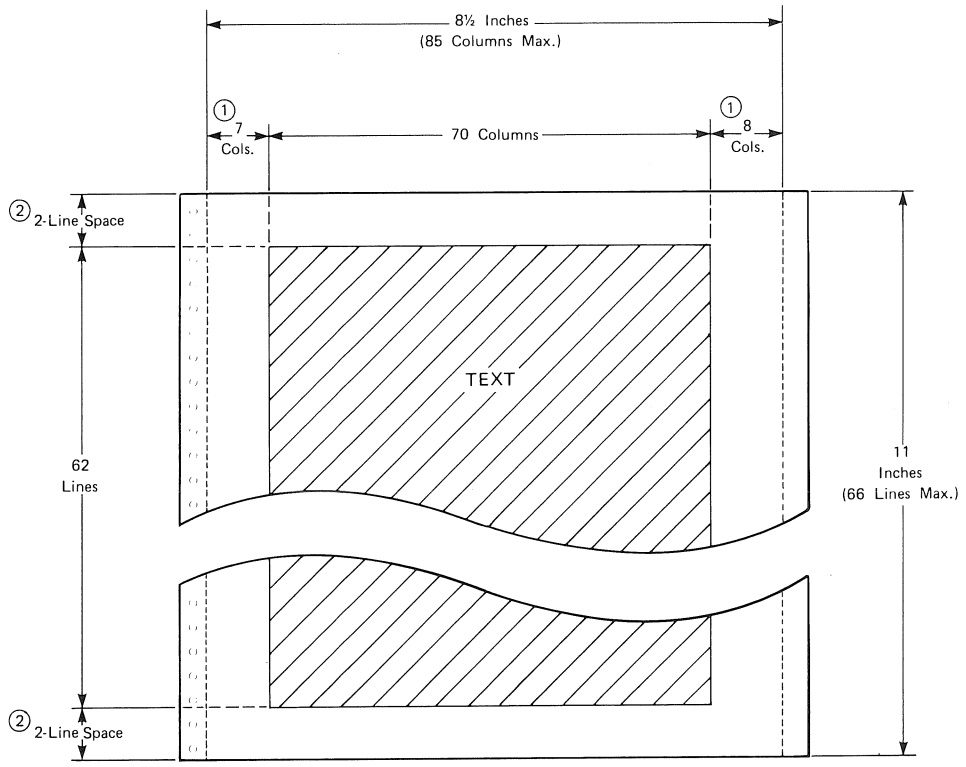


Figure 3-2 Text Layout (.STANDARD 60)

Understanding these relationships allows the user to more readily predict the number of actual lines of print that will appear on any given page. Knowing this, the user may then force appropriate page breaks that will affect more efficient formatting of the text.

This is especially important when print devices are used in which horizontal and vertical positioning is entirely arranged by the user (i.e., manually and via DSR commands).

For example, when using the DIABLO printer, a form feed switch is used to initially position the vertical feed mechanism. The top margin of the paper is then set by the user predicated on the maximum number of lines (in this case 66) allowed per page before a form feed again executes a page break (See Figure 3-3).



DIABLO PRINTER
.PAGE SIZE 62, 70

NOTES

① Mechanical Horizontal Paper Positioning (into left margin stop).
② Top and bottom page spacing set by user.

Figure 3-3 Diablo Printer Form Layout

Running Heads

Running heads consist of information and automatic line spacing for the top of each page. Depending on the format, there may be one or two lines of header information followed by two blank lines prior to the body of the text.

Running heads have several options. If .PAGING is enabled, the first line will consist of a page header positioned against the right margin, with a title (if included) positioned at the extreme left margin. In addition, if a subtitle is specified, a second information line will be used and the subtitle will be positioned directly below the title (See Figure 3-2).

3.1.1.1 .PAGE SIZE

This command is used at the beginning of a file to specify the maximum number of lines of text per page (n1) and the maximum width of the text in columns (n2).

The command may be formatted as follows:

```
.PAGE SIZE n1,n2

.PS n1,n2
```

Terminators:{.|EL|;|!}

Characteristics:

- The command initially causes a .BREAK and, if a .NO PAGING command has been entered, resumes the Paging Mode.
- The length parameter (n1) is optional with a minimum to maximum range of 13 to 9999.
- The width parameter (n2) is optional, with a minimum to maximum range of 3 to 150.
- If +n1 or +n2 is used, n1 or n2 is added to the current value.
- If -n1 or -n2 is used, n1 or n2 is subtracted from the current value.

Defaults:

If the command is not used, DSR will default to a line value (n1) of 58 and a column value (n2) of 60.

The default values are predicated on a standard typewriter paper size of 8 1/2 by 11 inches and the use of a line printer. As such, the values allow the text to accomodate the width of the paper in regard to its manual horizontal positioning, and the length of the paper in regard to the vertical page breaks (See Figure 3-1).

3.1.1.2 .NO HEADERS and .HEADERS ON

These commands allow the printing of running head information to be disabled and resumed.

The commands may be formatted as follows:

Disable	Resume
.NO HEADERS	.HEADERS ON
.NHD	.HD ON
	.HD

Terminators: {.|EL|;|!}

Characteristics:

If headers are enabled, the initial three or four lines of the text area, for every page, are reserved by the DSR for the insertion of one or two lines of header information, (i.e., title, page number, subtitle), followed by two blank lines.

However, if headers are disabled, the first printed line of the text area will appear at the top of the page, as specified by the print device (See Figure 3-2).

Under these conditions, the following arrangements are possible:

- If headers are enabled and paging is disabled (.NO PAGING), the entire document will consist of only one page. In this case, the first three lines of the text area will be blank, unless header information is forced by the use of a .TITLE and (if desired) a .SUBTITLE command, directly followed by a .FIRST TITLE command.

- If paging is enabled, the first header line of every page except the first will contain page information (page n or page n-n) positioned against the right margin. However, page information may also be placed on the first page by the singular use of a .FIRST TITLE command.

- If a .CHAPTER command is initially used, a title (from the chapter buffer) will appear at the left margin of every subsequent page, along with the page information.

- If chapters are not used, title and subtitle information for the first and second header lines of every subsequent page may be formed by the use of the .TITLE and .SUBTITLE commands, with the subtitle being directly positioned under the title. Again, the first page may be similarly formatted by the addition of a .FIRST TITLE command.

- If the user does not desire to be restricted to the default arrangements, header information may be re-formatted via a .LAYOUT command (refer to section 3.1.1.5).

Defaults:

The insertion of header information is initially enabled by default.

3.1.1.3 .HEADERS UPPER/MIXED/LOWER

These commands allow the header words "page" and "index" (if indexing is initiated) to appear in upper case (e.g., PAGE), lower case (e.g., page), or mixed case (e.g., Page). The effect of this command occurs at the start of the next page.

The commands may be formatted as follows:

```
.HEADERS UPPER

.HEADERS LOWER

.HEADERS MIXED
```

Terminators: {.|EL|;|!}

Defaults:

Initially, the header words are printed in mixed case by default (i.e., Page and Index).

3.1.1.4 .DATE and .NO DATE

These commands allow the printing of the current system date, under the number of each page-header, to be enabled or disabled. However, the .DATE command will only be effective if a .SUBTITLE command has been previously entered.

The date is printed in the Form: DD MMM YY.

The commands may be formatted as follows:

Enable	Disable
.DATE	.NO DATE
.D	.ND

Terminators: {.|EL|;|!}

3.1.1.5 .LAYOUT

This command causes a .BREAK, and re-arranges the header information (titles/subtitles and page numbers), for all subsequent pages, as specified by the user. Thus, the effect of this command occurs at the start of the next page.

The command uses two number-parameters (n1 and n2). The n1 parameter specifies the arrangement, while the n2 parameter is used to specify bottom-of-page spacing for page numbers.

Using n1, the following arrangements are possible:

1. Titles/subtitles may be made to appear at the top-center of a page with the page numbers centered at the bottom.

2. Titles/subtitles may be made to appear at the top-right of odd numbered pages and the top-left of even numbered pages with the page numbers centered at the bottom.

3. A standard page arrangement, with the inclusion of running page numbers that are bordered by hyphens (e.g., - 23 -) and centered at the bottom of the page. Unlike normal page numbering, that is affected by section commands (chapter, appendix, etc.), running numbers are consecutive page numbers sequentially applied to the entire document.

Using n2, n lines are reserved at the bottom of each page for the page number. The blank lines are not additionally allocated, but are taken from the text area specified by the .PAGE SIZE command. When n2 is used, at least n-1 blank lines will occur between the last line of text and the page number.

The command may be formatted as follows:

```
.LAYOUT n1, n2

.LO n1, n2
```

Terminators: {.|EL|;|!}

Characteristics:

- If n1 is a zero (0), n2 is not allowed and the standard page arrangement is used.
- If n1 is a one (1), the title/subtitle is centered and n2 is required.
- If n1 is a two (2), the title/subtitle is flush right (odd page) or left (even page) and n2 is required.
- If n1 is a three (3), n2 is required and standard page numbering is used along with running page numbers; the latter being centered at the bottom of the page.

3.1.2 Disable and Resume Paging Mode

For multi-paged documents, the Paging Mode provides sequential numbering (Page n) for every page except the first (i.e., 2, 3, 4, etc.). However, for section oriented documents (e.g., chapters), the numbers are prefixed (Page n-n) with an appropriate section number (e.g., Chapter 1-2, 1-3, 1-4, etc.)

3.1.2.1 .NO NUMBER and .NUMBER PAGE

The .NO NUMBER command disables page numbering (normal or running), while the counting of pages continues. Thus, if numbering is resumed by the insertion of a .NUMBER PAGE command, page numbers will re-appear that are sequentially correct. However, by using a count-parameter (c), with the .NUMBER PAGE command, and specifying the desired value in either digits or letters, a new sequence of page numbering can be forced.

If the count-parameter consists of a letter (i.e., a, b, etc), or letters (i.e., aa, ab, etc.), the new page number sequence will start with the numerically equivalent value of the letter (i.e., a through z = 1 through 26), or letters (i.e., aa through az = 27 through 52, etc.).

If the count-parameter consists of a digit, or digits, the new sequence will start with the numeric value specified. Moreover, signed numbers may be used to affect a relative change in the current page number by addition or subtraction.

The commands may be formatted as follows:

Disable	Resume
.NO NUMBER	.NUMBER PAGE c
.NNM	.NMPG c

Terminators: {.|EL|;|!}

Characteristics:

- If numbering is resumed and c is not used, the current value of the page counter will appear.
- If c consists of letters, the page counter is forced to the equivalent numeric value of c.

- If c consists of digits (maximum 9999), the page counter is forced to the value of c.

- If +c is used, c is added to the current value of the page counter.

- If -c is used, c is subtracted from the current value of the page counter; an error occurs if a negative value is produced.

Defaults:

The page numbering mode is initially enabled by default.

3.1.2.2 .NUMBER RUNNING

This command is used to specify and/or modify running page numbers. The parameters, restrictions, and characteristics are identical to those of the .NUMBER PAGE command (refer to section 3.1.2.1).

Running page numbers are initiated by a .LAYOUT 3 command (refer to section 3.1.1.5), and only decimal numbers can be displayed.

During the construction of an automatic table of contents and/or a two-column index the use of running page numbers may be indicated to allow page references to be made to the running numbers instead of section oriented numbers.

The command may be formatted as follows:

```
.NUMBER RUNNING c

.NMR c
```

3.1.2.3 .DISPLAY NUMBER

This command causes a .BREAK, and transforms page numbering into a format specified by the user (i.e., decimal, octal, hexadecimal, roman numerals, or letters). Moreover, the case (upper, lower, mixed) of all roman numerals or letters can be specified.

The command does not change values, but merely outputs equivalent values in the format specified.

To accomplish this, the user specifies a display-descriptor-parameter code (refer to section 2.1.3.7 for codes).

The command may be formatted as follows:

.DISPLAY NUMBER y

.DNM y

Terminators: {.|EL|;|!}

Characteristics:

- The .DISPLAY NUMBER command should be inserted following the page formatting commands the user desires to affect (e.g., .FIRST TITLE, .NUMBER PAGE, etc).

- If page letters are desired, the alphabetic characters equate with 1 through 26, repeated for each level (i.e., A - Z = 1 - 26, AA - AZ = 27 -52, BA - BZ = 53 - 78, etc.).

Examples:

entering:

```
.FIRST TITLE          !number first page.
.DISPLAY NUMBER LU    !display as upper-case letter.
```

produces:

Page A

entering:

```
.NUMBER PAGE 200      !re-number page decimally.
.DISPLAY NUMBER O     !display page number in octal.
```

produces:

PAGE 310

3.1.2.4 .NO PAGING and .PAGING

These commands respectively disable and resume the Paging Mode. When the Paging Mode is disabled, the remainder of the document, up to the next .PAGING command, becomes a single page. This means that the document will not be split into numbered pages and header spacing (refer to section 3.1.1.2) will not be reserved by DSR. It does not mean that the mechanical page breaks effected by certain print devices will be eliminated.

The commands may be formatted as follows:

Disable	Resume
.NO PAGING	.PAGING
.NPA	.PA

Terminators: {.|EL|;|!}

Defaults:

The Paging Mode is initially enabled by default.

3.1.3 Subpage Initiation and Numbering

When change-pages are inserted in a document, it is often desirable to be able to simply add a letter (A-Z) to the current page number that directly precedes the change-pages. When the additions have been defined in this manner, the termination of sub-paging returns page numbering to the normal mode.

3.1.3.1 .SUBPAGE and .END SUBPAGE

Both of these commands cause a .BREAK and start a new page, while they respectively initiate and terminate sub-paging. When sub-paging is initiated, a paging mode is effectively executed with an altered number scheme (Page n1 or Page n-n1). That is, the incrementation of page-numbers is suppressed and the letter sequence is appended to each change-page (e.g., page 2A, 2B, 2C, etc. or page 1-5A, 1-5B, 1-5C, etc.).

However, the number of change-pages per command is not restricted to 26 (A-Z), since the letters are re-sequenced for subpages occuring in excess of 26 (i.e., AA - AZ = 27 - 52, BA - BZ = 53 - 78, CA - CZ = 79 - 104, etc.).

When sub-paging is terminated, the normal page numbering scheme is resumed.

The commands may be formatted as follows:

Enable	Disable
.SUBPAGE	.END SUBPAGE
.SPG	.ES

Terminators: {.|EL|;|!}

3.1.3.2 .NUMBER SUBPAGE

This command can force a new sequence of subpage lettering by the use of a count-parameter (c).

If the count-parameter consists of a letter (i.e., a, b, etc.), or letters (i.e., aa, ab, etc.), the new subpage sequence will start with the letter(s) directly specified by c.

If the count-parameter consists of a digit, or digits, the new subpage sequence will start with a letter (i.e., 1 through 26 = a through z, etc.), or letters (i.e., 27 through 52 = aa through az, etc.), equated with the numeric value specified by c.

Signed numeric values may be used to affect a relative change in the current subpage letter(s) by the addition or subtraction of equated values.

The command may be formatted as follows:

.NUMBER SUBPAGE c

.NMSPG c

Terminators: {.|EL|;|!}

Characteristics:

- A parameter (c) must be specified with this command.
- If letters are used for c, the subpage letters are forced to c.

- If digits are used for c, the subpage letters reflect the equivalent numeric value.

- If +c is used, c is added to the current equated value of the subpage letter(s). For example, if the current subpage letter is "R" and plus one is specified (c = +1), the new subpage letter is "S".

- If -c is used, c is subtracted from the current equated value of the subpage letter(s). For example, if the current subpage letter is "R" and minus one is specified (n = -1), the new subpage letter is "Q".

- If -c is used, the resultant equated value of the new subpage letter(s) must not be a negative value.

3.1.3.3 .DISPLAY SUBPAGE

This command causes a .BREAK and transforms subpage numbering into a format specified by the user (e.g., decimal, octal, hexadecimal, roman numerals, or letters). Moreover, the case (upper, lower, mixed) of all roman numerals or letters can be specified.

The command does not change values, but merely outputs equivalent values in the format specified.

To accomplish this, the user specifies a display-descriptor-parameter code (refer to section 2.1.3.7 for codes).

The command may be formatted as follows:

```
.DISPLAY SUBPAGE y

.DSP y
```

Terminators: {.|EL|;|!}

Characteristics:

- The .DISPLAY SUBPAGE command should be inserted following the subpage formatting commands the user desires to affect (i.e., .SUBPAGE and .NUMBER SUBPAGE).

- If subpage letters are already invoked by a .NUMBER SUBPAGE command, a letter code (i.e., LU or LL) will merely affect the case of the letter(s).

3.2 TITLE FORMATTING COMMANDS

With headers and paging enabled by default, the first three or four lines on every page are reserved for running head information (title, page, and subtitle).

With this arrangement, page information is automatically provided for every page except the first, while the title formatting commands are used to provide page, title, and subtitle information for all pages (refer to section 2.1.1.2).

3.2.1 .FIRST TITLE

This command is used to force running head information to appear on the first page.

If the command is used alone, only the page information (i.e., Page 1) will appear. However, if the command is placed on the line following a .TITLE or the line preceding or following a .SUBTITLE command, all of the header information provided will also appear on the first page.

If a .CHAPTER command is used, DSR will arrange the chapter title on the first page without page information (i.e., Page One will not appear).

The .FIRST TITLE command must, however, precede any text input for the first page.

The command may be formatted as follows:

```
.FIRST TITLE

.FT
```

Terminators: {.|EL|;|!}

3.2.2 .TITLE

This command causes a .BREAK and takes the text-parameter (textl), directly following the keyword, as the title. The text will appear at the left margin and on the first header line of every page except the first; However, the title can appear on the first page if a .FIRST TITLE command is added.

The title may be re-arranged as specified by the .LAYOUT command (refer to section 3.1.1.5).

The command may be formatted as follows:

```
.TITLE textl

.T textl
```

Terminators: {EL}

Characteristics:

As shown, this command can only be terminated by a line feed; therefore, if a .FIRST TITLE command is also used it must be placed at the beginning of the next line.

Example:

```
.TITLE textlEL>

.FIRST TITLE{.|EL|;|!}
```

3.2.3 .SUBTITLE and .NO SUBTITLE

The .SUBTITLE command causes a .BREAK and takes the text-parameter (textl), directly following the keyword, as the subtitle. The text will appear at the left margin and on the second header line of every page except the first. However, the subtitle can appear on the first page if a .FIRST TITLE command is added.

As header information, the subtitle may be re-arranged as specified by the .LAYOUT command (refer to section 3.1.1.5).

The subtitle feature is disabled by entering the .NO SUBTITLE command.

The commands may be formatted as follows:

Enable	Disable
.SUBTITLE textl	.NO SUBTITLE
.ST textl	.NST

Terminators: {EL}

Characteristics:

This command can only be terminated by a line feed. Therefore, if a .FIRST TITLE command is also used it must be placed at the beginning of the next line.

Example:

```
.TITLE textl{EL}

.SUBTITLE textl{EL}

.FIRST TITLE{.|EL|;|!}
```

3.2.4 .NO AUTOSUBTITLE and .AUTOSUBTITLE

The .AUTOSUBTITLE command allows text for the subtitle information lines to be taken from each header-level-one entry used in the document. The feature is available by default and may be disabled by the .NO AUTOSUBTITLE command. However, the enabling command is only operable if a .SUBTITLE command has been previously entered.

The automatic header level feature is initiated by .HEADER LEVEL commands (refer to section 3.5.2.1). to provide up to six numbered levels (e.g., 1.1, 1.1.1, etc.) for sectioning and subsectioning documents and should not be confused with the running head information (i.e., page, title, and subtitle).

Since six numbered header levels can exist, an optional signed (+/-) number-parameter (n) is available with the .AUTOSUBTITLE command to specify a header level text other than the first.

The commands may be formatted as follows:

Disable	Resume
.NO AUTOSUBTITLE	.AUTOSUBTITLE n
.NAST	.AST n

Terminators: {.|EL|;|!}

Characteristics:

- If the .AUTOSUBTITLE command is used, it must be inserted before the .HEADER LEVEL command it is first desired to affect.

- If n is used, n will specify the common header level number from which the autosubtitles will be derived.

- If +n is used, n will be added to the current autosubtitle header level specification.

- If -n is used, n will be subtracted from the current autosubtitle header level specification.

- If n = 0, autosubtitling will be effectively disabled from that point on.

- If n = 99, autosubtitling will occur for all header levels.

- Autosubtitling will not work with the alternative header level 3-6 formats (refer to section 3.5.2.1).

Defaults:

- The autosubtitling feature is initially available by default for a header level of 1.

- The default value for n is the current autosubtitle header level specification.

Example:

In practice, .AST is inserted at the start of a sectioned document to provide header-level-one subtitles. However, the following example is used to demonstrate the interaction effected by the command:

```
.ST text 0              !subtitle is text 0.
   .
   .
.AST                    !value of n is 1.
   .
   .
.HL1 text 1             !subtitle is text 1.
   .
   .
.HL2 text 2
   .
```

```
   .
.AST+2                  !value of n is 3.
   .
   .
.HL3 text 3             !subtitle is text 3.
   .
   .
.AST-2                  !value of n is 1.
   .
   .
.HL4 text 4             !subtitle is text 1.
```

3.3 SUBJECT MATTER FORMATTING COMMANDS

The subject matter formatting commands are used to specify formatting and enhancements for the body of the text.

With these commands case and margin specifications are made, spacing is defined, and the shape of the text is specified along with character and word enhancements.

3.3.1 Case Specifications

Case specifications for individual letters and words can be made on any type of I/O terminal by using the appropriate single flag characters (i.e., UPPERCASE, LOWERCASE, and CAPITALIZE).

However, the results of invoking a case lock for the entire text, via flag pairs or commands (i.e., .UPPER CASE or .LOWER CASE), depends on both the selected directive and the type of I/O terminal used (i.e., Fixed-Case or Variable-Case).

For example, a lower case flag pair (\\) or command will ensure a lower case lock on any terminal. However, an upper case flag pair (^^) or command affects a no-case-change and will, therefore, only lock a terminal in its normal mode; that is, upper case for a fixed terminal and lower case for a variable terminal. Thus, the use of lower-case-lock directives will translate upper case input into lower case output, while the use of the upper-case-lock directives will <u>not</u> translate lower case input into upper case output.

3.3.1.1 .UPPER CASE

This command affects a no-case-change. Therefore, it will effectively lock a fixed case terminal in its normal upper case mode and a variable case terminal in its normal lower case mode. Thus, the command will not translate lower case input into upper case output.

The command may be formatted as follows:

```
.UPPER CASE

.UC
```

Terminators: {.|EL|;|!}

Characteristics:

- The .UPPER CASE command functions in the same manner as the UPPERCASE flag pair (^^).

- For a fixed case terminal, the command is used following a .LOWER CASE command to return the output to the normal upper case mode, as desired; thus affecting upper and lower case output via flag usage.

- For a variable case terminal the command has no practical value, since it has no effect on the output.

3.3.1.2 .LOWER CASE

This command effectively forces a lower case lock on a terminal, regardless of the type of terminal used. Thus, when this command is used, all affected letters in the source text will be output in lower case.

The command may be formatted as follows:

```
.LOWER CASE

.LC
```

Terminators: {.|EL|;|!}

Characteristics:

- With any terminal, the command functions in the same manner as the LOWERCASE flag pair (\\).

- With any terminal, all affected text will be output in lower case.

- When initially used with a fixed case terminal, the command will allow all of the text to be translated into lower case output. With this arrangement a mixed case document can be achieved as follows:

  1. Initiate the input file with a .LOWER CASE command.

  2. Specify letter by letter and word by word case requirements via the appropriate use of the flags.

  3. Print the output on a variable case printer.

3.3.2 Margin Specifications

The margin commands are used to continually adjust the width of the text to the requirements of the format. It should be noted, however, that these commands interact with many other commands (e.g., .LEFT, .RIGHT, .INDENT .PAGE SIZE, .CHAPTER, etc.).

Further, the margin commands initially evoke a .BREAK, which will output the current line with no justification and place the next word at the beginning of the next line.

The initial default value settings for the left and right margins are zero and sixty, respectively. The parameter (n) default value for the .LEFT MARGIN command is also zero, while the n default value for the .RIGHT MARGIN command is the page-width. Moreover, a .RIGHT MARGIN command setting can increase, but not decrease, the page-width.

3.3.2.1 .LEFT MARGIN

This command causes a .BREAK and then sets the left margin to the value of the number-parameter(n). The number used must be less than the right margin. In addition, the total value of any subsequent indent (.INDENT) plus the left margin must not be less than zero.

The command may be formatted as follows:

```
.LEFT MARGIN n

.LM n
```

Terminators: {.|EL|;|!}

Characteristics:

- Note, in regard to the print column (PC), that the value of n equals the number of spaces that directly precede the print column (e.g., LM0 = PC1, LM1 = PC2, etc.).
- If n is not used, the left margin default value is 0.
- If n is used, the left margin is forced to the value of n.

- If +n is used, n is added to the current value of the left margin; however, if the resultant value is greater than the right margin an error message will be issued by DSR.

- If -n is used, n is subtracted from the current value of the left margin; however, if the resultant value is less than zero an error message will be issued by DSR.

Defaults:

The initial left margin default value and the default value for n is zero.

3.3.2.2 .RIGHT MARGIN

This command causes a .BREAK and then sets the right margin to the value of the number-parameter(n).

The characters on a line will not extend beyond the right margin setting, unless they are extended via a .LITERAL command.

The command may be formatted as follows:

```
.RIGHT MARGIN n

.RM n
```

Terminators: {.|EL|;|!}

Characteristics:

- If n is not used, the right margin will default to the page-width.

- If n is used, the right margin is forced to the value of n.

- If +n is used, n is added to the current value of the right margin.

- If -n is used, n is subtracted from the current value of the right margin.

- If the new value of the right margin is greater than the original page width setting (.PAGE SIZE) the page width is reset to the new right margin value.

- If a .CHAPTER or .STANDARD command is issued, the right margin is reset to the page width value defined by the .PAGE SIZE command, otherwise the default value (60) is assumed.

Defaults:

The initial right margin default value is 60, while the default value for n is the page-width.

3.3.3 Fill and Justify

When DSR is processing the input file with the Fill and Justify features enabled, words are selected for a line of output until a word is reached that will exceed the right margin. At that point the space between the selected words is increased so that the last letter of the last word is exactly aligned with the right margin. This creates a vertically even right margin. To accomplish this, the Fill regards all real spaces (tabs excepted) and line feeds as normal word-space (unless .AUTOPARAGRAPH is enabled).

Where filling may result in abnormally large word-spacings, a .HYPHENATION command is available to affect the automatic hyphenation of words.

Both Fill and Justify are initially available by default, and either feature can be disabled (.NF and .NJ) and individually re-enabled (.F and .J) by command.

However, while the Justify can be independently disabled (.NJ) without affecting the Fill, the Fill cannot be independently disabled (.NF) without automatically disabling the Justify.

Therefore, in order to implement justification alone, the Fill must be disabled prior to re-enabling the Justify (i.e., .NF.J). A reversal of this order (.J.NF) would cause the last command to negate the effect of the first.

Finally, both commands initially evoke a .BREAK, which outputs the current line with no justification and places the next word at the beginning of the next line.

3.3.3.1 .NO FILL and .FILL

The .FILL command causes a .BREAK and causes each subsequent line of output to be filled. This means that words are added to the current line until the addition of another complete word would exceed the right margin. The command will also automatically re-execute the last .JUSTIFY or .NO JUSTIFY command entered.

The .NO FILL command disables both filling and justification. Moreover, the command also disables any .AUTOPARAGRAPH and/or .AUTOTABLE commands previously entered; a subsequent .FILL command will re-enable these commands.

The commands may be formatted as follows:

Disable	Resume
.NO FILL	.FILL
.NF	.F

Terminators: {.|EL|;|!}

Characteristics:

- If text is filled and not justified (.F.NJ) a ragged right is achieved. However, since the Fill regards all spaces and line feeds merely as word separators, the words following such spacing will be used as fill for the current line and the output will not mirror the input.

- If filling is disabled (.NF), justification is also disabled (automatically) and the output mirrors the input (i.e., all input spacing will be taken literally).

- If, for some special purpose (e.g., margin notes), justification alone is desired, the Fill must be disabled prior to enabling the Justify (.NF.J).

Defaults:

Fill and Justify are both initially enabled by default.

3.3.3.2 .NO JUSTIFY and .JUSTIFY

The .JUSTIFY command causes a .BREAK and then adjusts word spacing for all subsequent lines of text.

The .NO JUSTIFY command causes a .BREAK and disables justification.

The commands may be formatted as follows:

Disable	Resume
.NO JUSTIFY	.JUSTIFY
.NJ	.J

Terminators: {.|EL|;|!}

Characteristics:

Refer to the .FILL characteristics (3.4.2.1).

Defaults:

Justify and Fill are both initially enabled by default.

3.3.4 Vertical Spacing

These commands are used to specify and alter the vertical spacing between lines of text. However, due to the manner in which these commands interact the following definitions should be clearly understood:

- The term "line spacing" defines the normal spacing that can be observed between lines of printed text.

- The term "blank line" defines the vertical spacing created when a line is output to the printer with the text blanked out.

These definitions are important because both terms represent additive numeric values that are used to specify vertical spacing, and improper spacing will result if the meaning of these values is confused.

3.3.4.1 .BREAK

This command causes the current line to be output without being filled or justified.

Many DSR commands (e.g., .PAGE SIZE) evoke the line break function prior to the execution of their designated function. Each of these commands is described as such (i.e., ...causes a .BREAK etc.); however, the use of these commands can affect paragraph formatting (i.e., indentation), as described under "characteristics".

The command may be formatted as follows:

```
.BREAK

.BR
```

Terminators: {.|EL|;|!}

Characteristics:

- If a .BREAK command is inserted directly following an attempt to perform the function of an .AUTOTABLE, .AUTOPARAGRAPH, .PARAGRAPH, or .INDENT command, the effect of these commands (indentation) will be cancelled.

- All DSR commands that evoke a .BREAK, when inserted directly following an attempt to perform the function of an .AUTOPARAGRAPH command, will cancel the effect of this command.

- Most DSR commands that evoke a .BREAK, when inserted directly following an attempt to perform the function of a .PARAGRAPH or .INDENT command, will cancel the effect of these commands; the following are excepted:

```
.PAGE SIZE
.TITLE, .SUBTITLE
.LEFT, .LEFT MARGIN
.RIGHT, .RIGHT MARGIN
.FILL, .NO FILL, .JUSTIFY, .NO JUSTIFY
.SPACING, .SKIP, .BLANK, .PAGE, .TEST PAGE
```

3.3.4.2 .SPACING

This command causes a .BREAK and then evokes new vertical line spacing, for the text, as specified by the user via a number-parameter (n).

The range of values for the parameter (1 through 5) equates with the allowable spacing (i.e., 1 = single-spacing, 2 = double-spacing, etc.). However, since single-spacing is initially available (as with a typewriter) by default, the use of the command is only required if the user desires to increase the size of the spacing.

Due to the manner in which the .SPACING command interacts with other vertical spacing commands, it is important to understand that n defines the spacing that can be normally observed between printed lines of text, and should not be confused with the spacing that results when a line of text is blanked out.

The command may be formatted as follows:

```
.SPACING n

.SP n
```

Terminators: {.|EL|;|!}

Characteristics:

- If the command is not used single-spacing is assumed.
- If the command is used and n is not specified an error will occur.
- If the command is used, n must be a positive value within the range of 1 through 5.
- If n is greater than 5 an error message will occur.

Defaults:

Single-spacing is available by default.

3.3.4.3 .SKIP

This command causes a .BREAK and then inserts a number of blank lines in the output. The number of blank lines inserted is derived from the multiplication of an associated number-parameter (n), specified by the user, and the current line spacing value (i.e., 1-5).

However, in order for the user to predict the actual number of lines that will be skipped, two conditions must be considered: (1) The execution of a .SKIP causes an initial line break which invokes an additional spacing value (+ sp). (2) Any given number of lines (with text or blanked) will always be one less than the number of line spaces.

Therefore, the following algorithm must be used to define the total number of blank lines that will be skipped.

$$(sk * sp) + sp - 1 = nl$$

where:

sk	is the .SKIP value (n)
sp	is the .SPACING value (n)
+sp	is the line-space value adjustment
-1	is the line-space adjustment
nl	is the number of lines skipped

The following examples demonstrate the effect of the additive values in specifying the number of lines skipped when the values of sk (2) and sp (3) are swapped.

$$(sk2 \times sp3) + sp3 - 1 = 8nl$$

$$(sk3 \times sp2) + sp2 - 1 = 7nl$$

The command may be formatted as follows:

```
.SKIP n

.S n
```

Terminators: {.|EL|;|!}

Characteristics:

- If a .SKIP is entered so that it coincides with the generation of a new page (e.g., via the action of a .PAGE command or an exceeding of the .PAGE SIZE) the command will have no effect.

- If the value of n is such that the resultant value is larger than the number of lines and line-spacing currently available on the page, the effect of the skip will be terminated when the page break occurs.

- If -n is used the resultant negative value indicates how far up from the bottom of the page the next line of text will appear.

- If the initiation of a .FOOTNOTE conflicts with the execution of a skip, the .SKIP defines the bottom of the page as the line directly above the footnote.

Defaults:

The default value of n is 1.

3.3.4.4 .BLANK

This command causes a .BREAK and then leaves a number of lines blank, as specified by a number-parameter (n).

The command is similar to the skip and performs similar functions, such as: defining the bottom of a page for the next line, via a negative value (-n), and resolving a possible conflict with a .FOOTNOTE command.

Unlike the .SKIP command, however, the .BLANK command ignores line spacing values; therefore, a multiplication of values does not occur.

The command may be formatted as follows:

```
.BLANK n

.B n
```

Terminators: {.|EL|;|!}

Characteristics:

- If a .BLANK command is entered so that it coincides with the generation of a new page (e.g., via the action of a .PAGE command or an exceeding of the .PAGE SIZE) the command will have no effect.

- If the value of n is larger than the number of lines currently available on the page, the effect of the blanking will be terminated when the page break occurs.

- If -n is used, n specifies the number of blank lines, up from the bottom of the page, above which the next line of text will appear.

- If the implementation of a FOOTNOTE conflicts with the execution of the command, the .BLANK will define the bottom of the page as the line directly above the FOOTNOTE.

Defaults:

The default value of n is 1.

3.3.4.5 .PAGE

This command causes a .BREAK and then conditionally advances the number of lines required to start a new page. The condition for an advance is that the current page contain at least one line of text. Therefore, the command might be used following text to start a new page for a major header level (i.e., .HL1), but could not be used repeatedly to generate successive blank pages.

The command may be formatted as follows:

```
.PAGE

.PG
```

Terminators: {.|EL|;|!}

3.3.4.6 .TEST PAGE

This command causes a .BREAK and then predicates an advance to a new page on a comparison between the remaining number of lines available on a page and a required number specified by the user. An advance to a new page will occur only if there are fewer lines available than have been specified.

By testing the number of lines remaining on a page, the user may ensure that certain material (e.g., all items in a list) will appear on the same page.

The command may be formatted as follows:

.TEST PAGE n

.TP n

Terminators: {.|EL|;|!}

Characteristics:

- When the command is used, n is required and must be a positive value.
- If the number of lines specified by n are available on the current page a new page will not be evoked.

3.3.5 Horizontal Spacing

These commands are used to specify and alter the horizontal spacing within each line of text.

3.3.5.1 .CENTER

This command causes a .BREAK and centers the following text-parameter (textl) around a column that is specified by halving either the page-width or a number-parameter (i.e., page width/2 or n/2).

All centering is relative to the current page-width setting and is, therefore, independent of left and right margin settings. However, to center textl relative to margin settings, the required value of n may be derived by simply adding the current value of the left and right margin (LM + RM = n). For instance, to center textl relative to a left margin of 10 and a right margin of 50 requires that the text be centered around column 30; therefore, the value required for n is 60.

The command may be formatted as follows:

```
.CENTER n;textl

.CENTRE n;textl

.C n;textl
```

Note that the semi-colon is only required if textl is entered on the same line.

Terminators: {EL}

Characteristics:

- If n is not used current page-width value is divided by 2.
- If n is used, textl is centered on n columns divided by 2.
- If +n is used, n is added to the page-width and the resultant value is divided by 2.
- If -n is used, n is subtracted from the page-width and the resultant value is divided by 2.
- If a resultant value is greater than the page-width, or negative, an error will be issued by DSR.
- If textl is not used the default text is a single space.
- Textl can also be entered at the beginning of the next line.
- If the character-length of textl is greater than the page-width an error will occur.
- All flags encountered within textl will be applied.

Defaults:

- The default for n is the current page-width.
- The default for textl is a single space.

Example:

In this example, the command is input as follows and n is not used (page width = 65).

```
.C;MEMOS
```

producing:

```
0                          3                          6
1                          3                          5
|                          |                          |
|<---------------------- MEMOS ------------------------>|
```

In the example, the halving of an odd number of columns, for an odd number of letters, centers the middle letter. This is one of four possible combinations and the only combination in which the middle letter can be centered. For instance, if either the width or the number of letters in the example is even the word will be split around an approximate center.

The following algorithm is used to specify the number of spaces required to move the text to the approximate center.

$$(pw - nk) / 2 = s$$

where:

pw	is the resultant page-width
nk	is the number of characters
s	is the number of preceding spaces

3.3.5.2 .TAB STOPS

There are, initially, 32 tab stop settings available to the user by default. The default values occur in consecutive multiples of eight (i.e., 8, 16, 24,....256) and each setting may be sequentially evoked by successive depressions of the TAB Key.

The .TAB STOPS command allows an alternative value or list of values to be specified via the use of a signed (+/-) or unsigned number-parameter (n). Signed values can be used to add or subtract n from the corresponding default value, or current setting, while unsigned values of n can be used to affect a direct replacement.

Since each succeeding value of n is used to overide a corresponding default value, the parameter must be a number or a list of numbers of increasing value (i.e., n1, n2, n3,....n32).

However, although a maximum default value of 256 exists, the largest practical value is 149, based on the maximum allowable page-width of 150 columns. This value is one less than the maximum number of print columns because each tab stop setting specifies the number of character spaces that will precede its associated print column. For example, adding one to each tab stop default value will define its associated print column (i.e., 9, 17, 24, 32,.... 257).

Note that no validity checks are made by DSR on the use of parameters. Therefore, proper use of the command is the responsibility of the user.

The command may be formatted as follows:

.TAB STOPS n1,n2,...n32

.TS n1,n2,...n32

Terminators: {.|EL|;|!}

Characteristics:

- If a .TAB STOPS command is not entered, subsequent depression of the TAB key will invoke the default values.

- When a .TABS STOPS command is entered, subsequent depressions of the TAB key will evoke the alternative values (n1,n2,..etc.), until they are changed via the insertion of a new command.

- If a comma is used within a number list, in lieu of a number, the relative position of the comma will evoke the corresponding default value.

- If multiple tabs are to be used to input characters on the same line, and a comma is used to invoke a default value, the number directly following the comma must be greater than the invoked default value (by at least 2). If it is not, the input character will be shifted right by one extra column; this occurs because each character is output with at least one preceding space.

- If +n is used, n is added to the current value (set or default).

- If -n is used, n is subtracted from the current value (set or default).

- If the TAB key is depressed for a column that already contains text, or for which no value of n exists, only a single space will be output. This occurs when the user enters more text characters than there are columns existing between two successive tab settings; invalidating the subsequent setting.

Defaults:

There are 32 tab stop default values available in multiples of eight (i.e., 8, 16, 24, 32,.....256).

Example 1:

The following example depicts the relationship between default values and a number list. Notice that the n values overide the default values on a positional basis, while the extra commas invoke the default values in the same manner.

```
   8 16  24  32  40  48  56    default values
   | |   |   |   |   |   |
.TS3,5,  ,  26,  ,  49,  ,    tab stop command
   | |   |   |   |   |   |
   | |   |   |   |   |   |
   | |   |   |   |   |   |
   | |   |   |   |   |   |
   | |   |   |   |   |   |
   3 5   24  26  40  49  56    resultant tab stops
   | |   |   |   |   |   |
   4 6   25  27  41  50  57    specified print columns
```

Notice in the example, that following the first extra comma (used to evoke default value 24) the value of the next number (26) is greater than the preceding default value (24).

Example 2:

```
.TS+2           !add 2 to default of 8,
 .              !tab stop is 10 (col 11).
 .
.TS+8           !new tab stop is 18 (col 19)
 .
 .
.TS-7           !new tab stop is 11 (col 12)
```

3.3.5.3 .INDENT

This command causes a .BREAK and forces the next line of text to begin at a column defined by a number-parameter (n). As with the .LEFT MARGIN and .TAB STOP commands, n specifies a number of spaces to the right of the left margin that will precede the print column (n +1).

A negative value may also be specified, to allow the text to begin n spaces to the left of the left margin (with the exception of a left margin of zero). This effect is useful for such things as margin notes.

Be aware that .INDENT only affects the text that directly follows the command (i.e., on the same or the next line), and is propagated no further. Moreover, the indent effect can be cancelled if another command is inserted between .INDENT and the text. Therefore, the best practice is to use the command independently.

The command may be formatted as follows:

```
.INDENT n

.I n
```

Terminators: {./EL;/!}

Characteristics:

- If n is used, the current value of the left margin is added to n.

- If +n is used, the current value of the left margin is added to n.

- If -n is used, n is subtracted from the current value of the left margin. However, if the resultant value is negative an error will be issued by DSR.

- If n = 0, the current value of the left margin will be used.

- If the command is used and n is not used, the current value of the paragraph indent will be used.

Defaults:

When n is not used, the default is the current indent value (n) of the .PARAGRAPH command (refer to section 3.3.6.1).

Example 1:

entering:

```
.LM8.TS8
.BR;Textl                !Textl is at lm8.
.I-8;MARGIN Text2        !Text2 is at ts8.
.I-7;NOTE Text3          !Text3 is at ts8.
.BR;Text4                !Text4 is at lm8.
```

produces:

```
LM   0 1 2 3 4 5 6 7 8
COL  1 2 3 4 5 6 7 8 9
     | | | | | |     T e x t 1
     M A R G I N     T e x t 2
       N O T E       T e x t 3
     | | | | | |     T e x t 4
```

Example 2:

```
.P3                     !paragragh indent = 3.
  .
.LM10                   !left margin = 10
  .
.I2                     !left margin = 12
  .                     !for this line only.
  .
.I-4                    !left margin = 6
  .                     !for this line only.
  .
.I                      !left margin = 13
  .                     !for this line only.
  .
.I0                     !left margin = 10
  .                     !for this line only.
```

3.3.5.4 .LEFT

Although the keyword is different, the use and operation of this command is identical to that of the .INDENT command.

The command may be formatted as follows:

```
.LEFT n

.L n
```

3.3.5.5 .RIGHT

This command causes a .BREAK and then forces the last character of the text-parameter (textl), that directly follows the command, to be positioned a number of spaces to the left of the right margin by a value specified by a number-parameter (n).

Thus, unlike the commands that affect the left margin and define a print column for the first text character by specifying the number of spaces which precede it, the .RIGHT command defines a print column for the last text character by specifying the number of spaces that will follow it.

The .RIGHT command only affects the line that directly follows the command. Once the command is executed, the right margin will be returned to its current value.

The command may be formatted as follows:

```
.RIGHT n;textl

.R n;textl
```

Note that the semi-colon is only required if textl is entered on the same line.

Terminators: {.|EL|;|!}

Characteristics:

- If n is used, n is subtracted from the current value of the right margin.
- If the result of the subtraction is negative an error will be issued by DSR: meaning that DSR attempted to force the last character to the left of the left margin.
- If n is not used, the value of the right margin is used.
- If -n is used, subtraction produces an algebraic addition to the right margin, and the last character will exceed the margin by the value of n.

Defaults:

The default value for n is zero.

3.3.5.6 .NO PERIOD and .PERIOD

Following the punctuation character that appears at the end of a sentence or independent clause [i.e.,a period (.), semi-colon (;), colon (:), question mark (?), or exclamation mark (!)] a double-space is output by default.

However, the double-space will only occur if: (1) Fill is enabled and (2) the punctuation character is followed by either a space or an end-of-line code.

If it is desirable to overide the double-space effect, the user can use use an ACCEPT flag to force the punctuation character to be taken as a normal character (refer to section 4.1.3.1 for an example).

These commands allow the double-space feature to be disabled and re-enabled.

The commands may be formatted as follows:

Disable	Resume
.NO PERIOD	.PERIOD
.NPR	.PR

Terminators: {.|EL|;|!}

Defaults:

The double-space is available by default.

3.3.5.7 .NO SPACE

When the last character of the last word on a line is input, and an end-of-line code is entered, DSR automatically generates a single space as a word-separator. This separates the last character on the previous line from the first character on the next line.

The .NO SPACE command disables this operation, thereby preventing DSR from equating an end-of-line code with a space.

The command may be formatted as follows:

.NO SPACE

.NSP

Terminators: {.|EL|;|!}

Defaults:

An end-of-line space is initially available by default.

3.3.6 Paragraphing

These commands allow the paragraph form (i.e., indented first line) to be applied to the text in three ways:

- Independently, as the user desires, via .PARAGRAPH commands.

- Automatically, for the entire text (i.e., all lines staring with spaces or tabs), via an .AUTOPARAGRAPH command.

- Automatically, for consecutive lines of a table (i.e., lines not starting with spaces or tabs), via an .AUTOTABLE command.

3.3.6.1 .PARAGRAPH

This command causes a .BREAK and then forms an independent paragraph, using the text that directly follows the command.

The command has three optional number-parameters (n1, n2, n3):

1. The positive n1 value (paragraph-indent) specifies the number of indent spaces for the first word of the paragraph.
   The negative n1 value (negative-indent) can be used to specify spacing to left of the left margin for special formatting.

2. A positive n2 value (paragraph-spacing) may be any number(including 0) and specifies the number of blank lines that will occur between paragraphs. However, if .SPACING values (1 through 5) have been previously specified, the n2 values will interact with the spacing values (sp) in specifying the number of blank lines to be output.
   A negative n2 value (paragraph-skip) alternatively specifies the number of blank lines up from the bottom of the page upon which the first line of the paragraph will appear. In this regard, n2 functions in the same manner as a .SKIP command with a negative value (refer to section 3.4.3.3).

3. The n3 value (paragraph-test-page) specifies the number of lines that must be available on the page to fully accommodate the paragraph. Thus, n3 functions in the same manner as a .TEST PAGE command.

Since use of the parameters is optional their non-use evokes the following default values.

- The default value for n1 is either the current paragraph-indent-value or (if n1 has not been specified) a default value of five (5).

- The default value for n2 is either the current paragraph-spacing value or (if n2 has not been specified) a default value of one (1).

- The default value for n3 is either the current value of paragraph-test-page or (if n3 has not been specified) a default value of two (2).

The command may be formatted as follows:

```
.PARAGRAPH n1,n2,n3

.P n1,n2,n3
```

Terminators: {.|EL|;|!}

Characteristics:

- The positive n2 parameter (para-spacing) can be any positive number and has an initial default value of 1.

- If n2 is specified or its default value (1) is used, and an .SP value is not specified, the number of blank lines (bls) occuring will be directly equal to the value of n2 (i.e., n2 = bls).

- If n2 specifies a value of 0 or 1 (or a default value of 1) and .SP is specified the number of blank lines will be directly equal to the value of .SP (i.e., sp = bls).

- If n2 specifies a value of 2 or greater and .SP specifies a value of 2 through 5, the number of blank lines will be equal to the value of n2 multiplied by the value of .SP reduced by one (i.e., [n2 * sp] - 1 = bls).

- If the negative n2 parameter (paragraph-skip) is used, n2 will specify the number of blank lines up from the bottom of the page upon which the first line of the paragraph will appear.

- The n3 (para-test-page) parameter tests for a number of available lines, that is determined by the resultant value of n3 plus n2 (positive) plus 1 multiplied by the value of .SP (i.e., [n3 + n2 + 1] * sp).

- If n3 is set to a value of zero, the n3 default value will be overidden.

Defaults:

- The default for n1 is either the current n1 value or 5.

- The default for n2 is either the current n2 value or 1.

- The default for n3 is either the current n3 value or 2.

Example 1:

```
.P,2,0          !n1 = 5, n2 = 2, n3 = 0
  .             !n1 => para-indent 5 spaces
  .             !n2 => para-space 2 blank lines
  .             !n3 => para-test-page for 3 lines
.P              !uses same parameter values
```

Example 2:

```
.SP2            !set double-spacing
  .
  .
.P,2,0          !n2 => para-space 3 blank lines
                !n3 => para-test-page for 6 lines
```

3.3.6.2 .SET PARAGRAPH

This command allows the user to pre-define optional values (n1, n2, n3) for a .PARAGRAPH command (refer to section 3.3.6.1) without causing paragraphing to occur.

The command may be formatted as follows:

```
.SET PARAGRAPH n1,n2,n3

.SPR n1,n2,n3
```

Terminators: {.|EL|;|!}

Defaults: (same as .PARAGRAPH)

3.3.6.3 .AUTOPARAGRAPH and .NO AUTOPARAGRAPH

These commands enable and disable the autoparagraph mode. When the mode is enabled any subsequent blank line or any line starting with a space is considered as the starting point for a new paragraph. This allows paragraphs to be automatically formatted without the use of additional commands.

With auto-paragraphing enabled, the first word for each paragraph is indented five spaces from the left margin.

The commands may be formatted as follows:

Enable	Disable
.AUTOPARAGRAPH	.NO AUTOPARAGRAPH
.AP	.NAP

Terminators: {.|EL|;|!}

Characteristics:

- .AUTOPARAGRAPH is only effective where filling is enabled.
- .AUTOPARAGRAPH indents the first character of each paragraph 5 spaces (or nl value of current .PARAGRAPH command).
- Enabling .AUTOPARAGRAPH or .NO AUTOPARAGRAPH disables .AUTOTABLE.
- Enabling .AUTOTABLE and .NO AUTOTABLE disables .AUTOPARAGRAPH.

3.3.6.4 .AUTOTABLE and .NO AUTOTABLE

These commands enable and disable the autotable mode. When the mode is enabled any subsequent line without a leading space is considered as the start of a new paragraph. The first word of each line, thus considered, will be indented five spaces from the left margin.

The user should be aware that the auto-table effect can be temporarily cancelled if intervening commands (e.g., .SKIP, .PAGE, etc.) are entered at an auto-table starting point (i.e., a line with no leading space).

The commands may be formatted as follows:

Enable	Disable
.AUTOTABLE	.NO AUTOTABLE
.AT	.NAT

Terminators: .{.|EL|;|!}

Characteristics:

- .AUTOTABLE is only effective where filling is enabled.
- .AUTOTABLE indents the first character for a table by 5 spaces (or nl value of current .PARAGRAPH command).
- Enabling .AUTOTABLE and .NO AUTOTABLE disables .AUTOPARAGRAPH.
- Enabling .AUTOPARAGRAPH or .NO AUTOPARAGRAPH disables .AUTOTABLE.

3.3.7 Character and Word Enhancement

Excepting the bar feature, the operation of all of the other enhancement features (e.g., underlining, bolding) is available by default. Under these conditions, the following commands allow the default operations to be disabled and resumed and the bar feature to be enabled and disabled.

To affect the execution of these operations, note that an enabled feature requires the simultaneous recognition of its associated flag. In some cases flag recognition is available by default (e.g., underlining), and the operations may be implemented without the use of commands; otherwise the required flag must be enabled.

Thus, enabling the flag provides flag recognition (preventing the character from being taken as text), while a simultaneously enabled feature allows the operation to be performed.

3.3.7.1 .DISABLE and .ENABLE UNDERLINING

These commands disable and re-enable the underline feature which, along with the UNDERLINE flag (&), is initially enabled by default. Therefore, the operation may be performed for as long as both the flag and feature are enabled.

Underlining allows any character, word, group of words, or space, to be underlined (refer to sections 2.2.3.6 and 4.1.3.3 for examples).

The commands may be formatted as follows:

Disable	Resume
.DISABLE UNDERLINING	.ENABLE UNDERLINING
.DUL	.EUN

Terminators: {.|EL|;|!}

Defaults:

Both the underline feature and the flag are initially available by default.

3.3.7.2 .NO HYPHENATION and .HYPHENATION

These commands disable and re-enable the hyphenation feature. However, since the feature is initially enabled by default and the HYPHENATION flag (=) is not, enabling recognition of the flag (.FLAGS HYPHENATE) will allow the operation to be performed for as long as both flag and feature are enabled.

Hyphenation may be used to close up excessively long spacing between words, which often occurs when margins are narrow and the line is interspersed with several long words (refer to section 4.1.3.6 for example).

The commands may be formatted as follows:

Disable	Resume
.NO HYPHENATION	.HYPHENATION
.NHY	.HY

Terminators: {.|EL|;|!}

Defaults:

The hyphenation feature is initially enabled by default; however, the flag is not.

3.3.7.3 .DISABLE BOLDING and .ENABLE BOLDING

These commands disable and re-enable the bolding feature. However, since the feature is initially enabled by default and the BOLD flag (*) is not, enabling recognition of the flag (.FLAGS BOLD) will allow the operation to be performed for as long as both flag and feature are enabled.

Bolding enhances the reader's awareness of a character, word, or group of words, by printing the affected characters darker than the rest of the text.

The commands may be formatted as follows:

Disable	Resume
.DISABLE BOLDING	.ENABLE BOLDING
.DBO	.EBO

Terminators: {.|EL|;|!}

Defaults:

The bolding feature is initially enabled by default; however, the flag is not.

3.3.7.4 .DISABLE and .ENABLE OVERSTRIKING

These commands disable and re-enable the overstrike feature. However, since the feature is initially enabled by default and the OVERSTRIKE flag (%) is not, enabling recognition of the flag (.FLAGS OVERSTRIKE) will allow the operation to be performed for as long as both flag and feature are enabled.

Overstrike is used to create single characters, that are not available on the keyboard, by overstriking two characters that are available. Such as, overstriking a seven (7) with a hyphen (-) to create a European 7 (i.e., 7̶).

The commands may be formatted as follows:

Disable	Resume
.DISABLE OVERSTRIKING	.ENABLE OVERSTRIKING
.DOV	.EOV

Terminators {.|EL|;|!}

Defaults:

The overstike feature is initially enabled by default; however, the flag is not.

3.3.7.5 .ENABLE, .DISABLE, .BEGIN, and .END BAR

The bar feature is used to insert a bar (|) in column one of a line to indicate to the reader that the text has been replaced or otherwise altered.

These commands enable the feature (.ENABLE BAR), specify where the change-bars will begin (.BEGIN BAR) and end (.END BAR), and disable the feature (.DISABLE BAR). The commands are used in the order described.

Once the feature is enabled, the begin and end commands are used (like flags) to initiate and terminate the insertion of change-bars. Thus, a bar will appear in column one of each successive line encompassed by the begin and end commands.

The commands may be formatted as follows:

```
.ENABLE BAR
.EBB

.BEGIN BAR
.BB

.END BAR
.EB

.DISABLE BAR
.DBB
```

Terminators: {.|EL|;|!}

3.4 SPECIAL INSERTION COMMANDS

The special insertion commands allow the user to introduce such things as figures, lists, notes and footnotes into the text. The commands also provide for the insertion of user designed information (e.g., block or flow charts) that is not subjected to DSR formatting; this information is, therefore, taken "as is".

3.4.1 User Designed Insertions

With the design insertion commands (i.e., .FIGURE, .LITERAL, .REPEAT) the user can respectively:

1. Reserve space on a current or subsequent page for the insertion of an externally produced (e.g., manually drawn) figure.

2. Literally reproduce text and/or graphic input in the output. With this feature the input format is user designed and virtually independent of DSR formatting.

3. Replicate a string of input characters, a specified number of times, in the output. With this feature the user can "draw" vertical or horizontal lines, using a specified character, or generate multi-character repetetive patterns.

3.4.1.1 .FIGURE and .FIGURE DEFERRED

These commands cause a .BREAK and reserve a number of blank lines, specified by a positive number-parameter (n), for the insertion of a figure; the lines may be reserved for a current (.FIGURE) or later (.FIGURE DEFERRED) page. However, the user cannot specify a number of blank lines that is greater than the number of lines available per page (i.e., via default or .PAGE SIZE command).

If the .FIGURE command is used and the required number of lines are not available, DSR will skip to the top of the next page and generate the specified number of lines. Thus, the .FG command can be used to unconditionally provide a specified number of blank lines at the top of a page.

If the .FIGURE DEFERRED command is used and the required lines are not available, the number is added to the current figure deferred value. Moreover, unlike the .FG command, if the desired space is not available on the current page the page will be filled before skipping to the top of the next page.

Either command may be used within a footnote. However, if the required lines are not available the skip to the next page, incurred by the .FIGURE command, will produce an error message and values cannot be accumulated for the .FIGURE DEFERRED commmand; thus, both commands are equivalent within a footnote.

The commands may be formatted as follows:

.FIGURE n

.FG n

.FIGURE DEFERRED n

.FGD n

Terminators: {EL/;}

3.4.1.2 .LITERAL and .END LITERAL

These commands initiate and terminate a literal block of information (i.e., text or a graphic display), the length of which may be optionally defined by a positive number-parameter (n).

The initiating command causes a .BREAK and provides the output with an exact copy of the input; this continues until the terminating command is recognized.

Within a literal block, the execution of all DSR commands and previously specified case, fill and justify operations is disabled. However, the affects of the following parameters are maintained: the current left margin setting; the current tab stop settings; the current spacing value; also , a right margin value of 150 is evoked.

Underlining and/or bolding can be evoked for the entire block. To effect underlining or bolding, the appropriate initiating flags are inserted on the line directly preceding the .LITERAL command, while the terminating flags are inserted directly following the .END LITERAL command. The affect of the flags on the literal text will, therefore, be inclusive and not selective.

The commands may be formatted as follows:

Enable	Disable
.LITERAL n	.END LITERAL
.LT n	.EL

Terminators: {.|EL|;|!}

Example:

The .TAB STOPS example, depicted in this manual (III.3.5.2), is a literal block design.

3.4.1.3 .REPEAT

This command allows a maximum of 50 characters, defined by a quoted-string parameter (q), to be printed a specified number (n) of times.

When the command is entered, the quoted-string characters must be contained within either single ('q') or double ("q") quotes.

The command may be formatted as follows:

.REPEAT n,"q"

.RPT n,"q"

Terminators: {.|EL|;|!}

Characteristics:

- If the .REPEAT command is entered with fill enabled a horizontal replication of the quoted-string will occur, line by line, the number of times specified by n.
- If the .REPEAT command is entered with fill disabled a vertical replication of the quoted-string will occur the number of times specified by n.
- The quoted-string is effectively treated as a new line of text; therefore, all flags and enhancements are operable within the text.

Example:

In this example the .REPEAT command is used to highlight commentary appearing in a functional specification.

entering:

```
.LM0.RM70
   .
   .
.RPT35"*/"
.B2
####This commentary is highlighted by a border.
.B2
.RPT35"*/"
```

produces:

```
*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/


     This commentary is highlighted by a border.


*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/
```

3.4.2 List and List Element Insertions

With the list formatting commands the user can automatically insert a sequentially numbered list within the text. The feature also has options which allow the user to alter the spacing between items (list elements) and create a bulletted instead of a numbered list.

Lists may be nested (a list within a list) in several ways. For instance, a numbered or bulleted list may be indented and nested within a main listing to specify a numbered subsequence or bulleted minor items relating to a major item.

Using list-level and list-element counters, DSR provides for the nesting of up to five lists. With these facilities, conflicts between nested levels cannot occur when parameter default values are invoked.

3.4.2.1 .LIST and .END LIST

These commands initiate and terminate the formatting of a list, the items of which are itemized by the use of .LIST ELEMENT commands.

An optional positive number-parameter (n) allows the user to specify blank lines of spacing before the first and between each subsequent line of the list. The number of blank lines (bls) that will occur is equal to the value of n (maximum 5) multiplied by the current .SPACING value (i.e., n * sp = bls). However, if n is equal to zero only the current line spacing value will be invoked.

An optional quoted-string parameter (q) allows the user to specify a common header, for each line, with a single character (e.g. bullet) other than the sequential numbers (1, 2, 3, etc.) available by default.

If a quoted-string character is used it must be contained within single ('q') or double ("q") quotes.

The commands may be formatted as follows:

Enable	Disable
.LIST n,"q"	.END LIST
.LS n,"q"	.ELS

Terminators: {.|EL|;|!}

Characteristics:

- Before executing the first .LIST ELEMENT command, the .LIST command performs a .TEST PAGE using the current paragraph-test-page value + 2 (refer to section 3.3.6.1).

- If the left margin is currently zero, the .LIST command will re-set the margin to a value of 9. This allows the first character of the text items (list elements) to be placed in column 10 (left margin + 1).

- If the current left margin is other than zero, the .LIST command will add a value of 4 to the left margin to affect the placement of the first character of the text items in a column that is equal to the resultant left margin + 1.

- Any fill, justify, case, margin, and spacing changes made to accommodate the elements of a list are restored by the .END LIST command.

Defaults:

- The default for n is the current paragraph-spacing value (refer to section 3.3.6.1) or one blank line.

- The default for q is a sequentially numbered sequence (i.e., 1, 2, etc.).

3.4.2.2 .LIST ELEMENT

This command appears on the line following the .LIST command and is used to introduce the text for each item in the list.

The command causes a .BREAK and then formats the item number or character for the associated line(s) of text.

The command is formatted as follows:

```
.LIST ELEMENT

.LE
```

Terminators: {.|EL|;|!}

Characteristics:

- If the left margin is currently set to zero, and a numbered list (e.g., 1, 2, etc.) is to be formatted, the command will add a value of 5 to the left margin to affect the placement of the numbers in column 6. A period is then inserted in the next column, for each number, followed by two real spaces and then the text (first character in column 10). For example:

  1. The first text character ("T") is in column 10.

- If the left margin is set to zero, and a character list (e.g., bullets) is to be used, the command will add a value of 6 to the left margin to affect the placement of the character (i.e., bullet), for each item, in column 7. Since periods are not required, each character is followed by two real spaces and then the text (first character in column 10). For example:

  - The first text character ("T") is in column 10.

- If the left margin is other than zero, and a numbered list is to be formatted, the numbers will appear in the column equal to the left margin + 1, followed by the periods, double-spaces, and text.

- If the left margin is other than zero, and a character list is to be formatted, the characters will appear in the column equal to the left margin + 2 (since periods are not required), followed by the double-spaces and the text.

Examples:

For the following three examples, .SPACING is equal to the default value of 1.

In the first example, the numbered items are single-spaced (.LS0).

In the second example, the items are bulletted and separated by two blank lines (.LS2"o").

In the third example, two lists with separate item headers, spacing, and margins, are nested (.LM0.LS2 and .LM9.LS0"*") and simultaneously terminated (.ELS.ELS).

entering:

```
.LM0.LS0

.LE;item 1

.LE;item 2

.ELS
```

produces:

```
1.  item 1          !"i" is in column 10 (LM0 + 9).
2.  item 2
```

entering:

```
.LM0.LS2"o"

.LE;item 1

.LE;item 2

.ELS
```

produces:

```
o  item 1           !"i" is in column 10 (LM0 + 9).

o  item 2
```

entering

```
.LM0.LS2
.LE;item 1
.LE;item 2
.LM9.LS0"*"
.LE;item 2a
.LE;item 2b
.ELS.ELS
```

produces

```
1.  item 1          !"i" is in column 10 (LM0 + 9).

2.  item 2
      *  item 2a     !"i" is in column 14 (LM9 + 4).
      *  item 2b
```

3.4.2.3 .NUMBER LIST

This command is used to alter the numbering sequence (e.g., from 1, 2, etc., to 4, 5, etc.) in an entire list or in any nested portion. The command also re-enables page numbering.

The command uses two number-parameters (n1 and n2) to affect alteration. The first number (n1) is used to specify one of five list-level counters (which equate with five possible levels of nesting). The second parameter (n2) is used to alter the sequence of the list-element numbers within the level specified by n.

The parameters may be signed (+/-) or unsigned numbers. Therefore, direct values may be assigned for the counters or specified values may be arithmetically added to current levels. However, the resultant value for a list-level counter must not be less than 1 or greater than 5, and the resultant value of a list-element counter must not be a negative number.

The command may be formatted as follows:

```
.NUMBER LIST n1, n2
.NMLS n1, n2
```

Terminators: {.|EL|;|!}

Defaults:

The default for n1 is the current list-level value, and the default for n2 is the current list-element value.

Example:

In the example, items 4 and 5 of a sequence are inserted within a page of text along with subsequence items 5a and 5b.

entering:

```
.LM0.LS2
.NMLS1,4                    !start list 1 at count of 4.
.LE;item 4
.LE;item 5
.LM9.LS                     !start list 2 at count of 1.
.LE;item 5a
.LE;item 5b
.ELS.ELS
```

produces:

```
4.  item 4                  !"i" is in column 10 (LM0 + 9).

5.  item 5

        1.  item 5a         !"i" is in column 14 (LM9 + 4).
        2.  item 5b
```

3.4.2.4 .DISPLAY ELEMENTS

This command causes a .BREAK and then formats list elements as specified by the user.

The command increases list element formatting possibilities (from decimal numbers and bullets) to octal, hexadecimal, roman numerals, and letters. Moreover, the case (upper, lower, mixed) of all roman numerals or letters can be specified. In addition, itemization symbols can be enclosed (e.g., in parenthesis or brackets) as the user desires.

To accomplish this, the command uses two parameters, arranged as follows:

1. A quoted-string-parameter (q) first specifies the character desired for the left closure (e.g., a left bracket "[").

2. A display-descriptor-parameter (y) code is then used to specify the form and case (refer to section 2.1.3.7 for codes).

3. A second quoted-string-parameter is then used to specify the character desired for the right closure (e.g., right bracket "]").

For example, to enclose sequential, lower-case, letters in parenthesis (i.e., (a), (b), etc.), the parameters are entered as follows:

```
"(",LL,")"
```

The command may be formatted as follows:

```
.DISPLAY ELEMENTS "q", y, "q"
.DLE "q", y, "q"
```

Terminators: {.|EL|;|!}

Characteristics:

- If a space is inserted between quotes (e.g., " "), the space will considered as a closure character:

```
.DLE " ",LL," " or .DLE ' ',LU,' '
```

- If closures are not desired, adjacent quotes can be entered back-to-back ("" or ''), with no intervening space:

```
.DLE "",d,"" or .DLE '',ru,''
```

- If a comma is inserted in place of a parameter, the current value will be invoked (if no value exists the parameter will be ignored).

- If bullets have been specified by a .LIST command, any format specified by the .DISPLAY ELEMENTS command will override them.

- The .DISPLAY ELEMENT command must be entered between the associated .LIST command and the first .LIST ELEMENT command.

Example:

entering:

```
.LM0.LS2
.LE;item 1
.LE;item 2
.LM9.LS0
.DLE"(",LL,")"
.LE;item 2a
.LE;item 2b
.ELS.ELS
```

produces:

1. item 1

2. item 2

   (a) item 2a

   (b) item 2b

3.4.3 Note and Footnote Insertions

With the following commands the user can:

1. Insert a formal note within the text, that will be automatically titled, spaced, and centered.

2. Insert a footnote, that may be formatted as the user desires, at the bottom of the page.

3.4.3.1 .NOTE and .END NOTE

These commands initiate and terminate the formatting of a note. When the initiating command is executed, a .TEST PAGE is initially performed, using the current paragraph-test-page value + 4, to determine if there is enough room on the page to accommodate the note. If sufficient space is not available the note will be output at the top of the next page.

The initiating command then causes a .BREAK, performs a .BLANK 2 multiplied by the .SPACING value (i.e., .B2 * .SPn = bls), and outputs the title (textl) in the center of the next line. If a title is not entered, the capitalized word "NOTE" is output by default.

Following the title, a .SKIP 1 is executed and the text is centered by executing a .LEFT MARGIN +15 and a .RIGHT MARGIN -15, if the current .LM setting is equal to zero; otherwise, the margins are set to +4 and -4. The lines of text are then filled and justified. Understand, that the aformentioned margins are default settings and the user may set the margins to any desired value.

When the termination command is executed, another .B2 * .SPn is performed to provide uniform blank line spacing (before and after the note) within the general text; the fill, justify, case, margin and spacing values are then restored.

The commands may be formatted as follows:

Enable	Disable
.NOTE textl	.END NOTE
.NT textl	.EN

Terminators:

.NOTE may only be terminated by a EL.

.END NOTE may be terminated with {.|EL|;|!).

3.4.3.2 .FOOTNOTE and .END FOOTNOTE

These commands allow a number of lines to be reserved at the bottom of the current page (or the next page if room is not available) for footnote information. The text directly following the initiating command is the first line collected for the footnote buffer, while the line prior to the terminating command is the last.

Besides text, the collected information may also contain any commands (e.g., .CENTER, .SKIP, etc.) and/or flags necessary to properly format the footnote. However, commands cannot be used that would affect or exceed the basic values used to shape the document.

Groups of commands that cannot be used to format a footnote are: Page (3.1), Title (3.2), and Section Formatting Commands (3.5). The exceptions within this group are the indexing commands (.INDEX, .SUBINDEX, .ENTRY). The remaining commands may be used, with the following exceptions:

.PAGE and .TEST PAGE
.FOOTNOTE
.STANDARD and .VARIABLE

The spacing reserved for the footnote is the product of an optional, positive number-parameter (n) multiplied by the current .SPACING value (refer to section 3.3.4.2).

The commands may be formatted as follows:

Enable	Disable
.FOOTNOTE n	.END FOOTNOTE
.FN n	.EFN

Terminators:

.FOOTNOTE may only be terminated with {;/EL}

.END FOOTNOTE may be terminated with {.|EL|;|!}

Characteristics:

- If n is not used, the required number of lines is computed by DSR.
- If n is used, its value must be equal to the number of output lines required to accommodate the footnote.

- If multiple footnotes are inserted for a single page, the space reserved at the bottom of the page will be equal to the sum of the values of n multiplied by the spacing value (i.e., n1 + n2 ... + n * sp = ls).

- When .FOOTNOTE is terminated by a semi-colon or a line feed, commands may be inserted to format and enhance the subject matter.

- When a footnote is initiated, current format values (e.g., case, fill and justify, spacing, margins, etc.) are saved, and restored when the footnote is terminated. Thus, any value changes invoked for the footnote are temporary.

3.5 SECTION FORMATTING COMMANDS

A sectioned document may consist of subsectioning for either a section-oriented manual or a chapter-oriented manual, one or more appendices, and an index.

For each major-section (section or chapter, appendix, index), DSR provides up to six levels of subsectioning via automatically sequenced header level commands.

Each header level within a major-section shares a common prefix. The prefix for a section or a chapter is the number of the section or chapter. The prefix for an appendix is the identifying letter (A, B, C, etc.), while the prefix for an index is the word "Index".

In order to affix an appropriate prefix for subsectioning, each major section must be explicitly defined. To accomplish this, chapters, appendices, and indices, are specified as major-sections by command prior to the insertion of their subsectioning commands. However, no major-section command is needed to produce a section-oriented manual. This type of manual is produced by the mere insertion of the header level commands. With this arrangement, automatic numbering will impose a section-oriented numbering system as opposed to a chapter-oriented system (e.g., level ones will be 1.0 and 2.0 instead of 1.1 and 2.1).

Since the creation of an index can be a complicated process, an entire chapter is devoted to the subject (refer to Chapter 5 CREATING AN INDEX).

3.5.1 Chapters and Numbering

A chapter is a major section defined as a chapter by command. The number of a chapter, whether imposed by DSR or the user, becomes the prefix for all of the header levels used within the chapter.

3.5.1.1 .CHAPTER

This command causes a .BREAK and initiates a chapter, taking the text-parameter (textl), directly following the keyword, as the title. The numbering of chapters will be automatic and sequential, starting with Chapter 1.

The command is formatted as follows:

```
.CHAPTER textl

.CH textl
```

Terminators: {.|EL|;|!}

Characteristics:

1. When the command is executed, a new page is automatically invoked and 12 blank lines are inserted.

2. The left margin is set to zero, and fill, justify, and single-spacing are enabled.

3. In upper case letters, the word "CHAPTER", followed by a space and an appropriate number (i.e., 1, 2, 3, etc.), is centered on the thirteenth line.

4. One blank line is then inserted, followed by the centered title. The title will be printed entirely in upper case, unless flags in the text specify otherwise.

5. Three blank lines are then inserted. Thus, the first line of the body of the text (i.e., Header Level One) will appear on the nineteenth line of the page (See Figure 3-2).

3.5.1.2 .NUMBER CHAPTER

This command is used to specify a chapter number for the next .CHAPTER command, following which the sequential numbering of subsequent chapters is resumed at the new level. The command also re-enables page numbering.

A count-parameter (c) is used with the command to specify the chapter number via digits or letters.

If the parameter consists of a letter or letters, the new chapter number sequence starts with the numerical equivalent of the letter (i.e., a through z = 1 through 26), or letters (i.e., aa through az = 27 through 52, etc.).

If the parameter consists of a digit or digits, the new chapter number sequence starts with the numeric value specified. Signed numbers may also be used to affect a relative change to the chapter numbers by addition or subtraction.

The command is formatted as follows:

.NUMBER CHAPTER c

.NMCH c

Terminators: {.|EL|;|!}

Characteristics:

- If c consists of letters, the chapter number is changed to the equivalent numeric value of c.
- If c consists of digits, the chapter number is changed to the value of c.
- If +c is used, add c to the current value of the chapter number.
- If -c is used, subtract c from the current value of the chapter number.
- If an .APPENDIX command directly follows the .NUMBER CHAPTER command, the word "APPENDIX" will be changed to "CHAPTER" and the specified number will be affixed; the reverse occurs if a .CHAPTER command directly follows a .NUMBER APPENDIX command (refer to section 3.5.3.2).

3.5.1.3 .DISPLAY CHAPTER

This command causes a .BREAK and then transforms chapter numbers into a format specified by the user (i.e., decimal, octal, hexadecimal, roman numerals, or letters). Moreover, the case (upper, lower, mixed) of all roman numerals or letters can be specified.

The command does not change the values, but merely outputs equivalent values in the format specified.

To accomplish this, the command uses a display-descriptor-parameter code (refer to section 2.1.3.7 for codes).

The command may be formatted as follows:

.DISPLAY CHAPTER y

.DCH d

Terminators: {.|EL|;|!}

Characteristics:

The .DISPLAY CHAPTER command should appear at the top of a page, following the chapter commands the user desires to affect (i.e., .CHAPTER and .NUMBER CHAPTER).

Example:

entering:

```
.NUMBER CHAPTER 2     !re-number chapter decimally.

.DISPLAY CHAPTER RU   !display upper case roman numeral.
```

produces:

CHAPTER II

3.5.2 Subsections and Numbering

Via Header Level commands (.HL1 through .HL6), DSR provides for the entry of up to six numbered subsections within a major section (Section, Chapter, Appendix, Index).

By specifying the level of a subsection, each command automatically evokes a sequential number that is prefixed to reflect the major section in which it appears. However, by using a numbering command, the automatic generation of a number may be disabled and a value (up to six digits) can be specified by the user.

For output formatting, DSR recognizes two types of header levels (non-run-in and run-in). A non-run-in header level (1 and 2) appears above the information, separated by a blank line. A run-in header level (3 through 6) runs into the first line of information, from which it is separated by a hyphen (-) and a space.

The following examples depict the manner in which automatic header levels are formatted by DSR. For all the examples, assume that the titles have been input in lower case.

1.1 HEADER LEVEL 1 FORMAT

The text is below the title and the title is automatically capitalized.

1.1.1 Header Level 2 Format

The text is below the title and the first letter of each word of the title is automatically capitalized.

1.1.1.1 Header Level 3 Format - The text immediately follows the title and the first letter of each word of the title is automatically capitalized.

3.5.2.1 .HEADER LEVEL

This command causes a .BREAK, executes a .TEST PAGE (using the current paragraph-test-page value + 7), and specifies both the header level (n) of the subsection and the text (text2).

The command is formatted as follows:

.HEADER LEVEL n text2

.HL n text2

Terminators: {.|EL|;|!}

Characteristics:

- The value of n may be any single digit from 1 through 6.
- If n is used, the header level is forced to the value of n.
- If +n is used, n is added to the last header level value used.
- If -n is used, n is subtracted from the last header level value used.
- If the header level is a 1, all text2 letters are printed in upper case; otherwise, only the first letter of each text2 word is printed in upper case.
- When a header level command is inserted, Fill and Justify are automatically enabled.
- If header levels are not run-in (e.g., 1 and 2), one blank line will follow the subsection (regardless of whether information is included).

- If header levels are run-in (e.g., 3 through 6), three blank lines will follow the inserted information; if no information is inserted, one blank line will follow.

- If the autosubtitle feature is enabled, and the header level is a 1, and a .SUBTITLE command was previously used, then text2 of the .HL command will be taken as the new subtitle for the running head information line (refer to .AUTOSUBTITLE, section 3.2.4).

Defaults:

If n is not used, DSR will default to the last header level value used.

Example:

This example depicts the manner in which header level outputs are automatically numbered for each major section.

	SECTION 1	CHAPTER 1
.hl1	1.0	1.1
.hl2	1.1	1.1.1
.hl3	1.1.1	1.1.1.1
	INDEX	**APPENDIX A**
.hl1	Index.1	A.1
.hl2	Index.1.1	A.1.1
.hl3	Index.1.1.1	A.1.1.1

3.5.2.2 .NUMBER LEVEL

This command re-enables page numbering and changes the number of the header level specified by n (n1 through n6). The sequential numbering of subsequent header levels is then resumed at the new level.

The command is formatted as follows:

```
.NUMBER LEVEL n1,n2,....n6

.NMLV n1,n2,....n6
```

Terminators: {.|EL|;|!}

Characteristics:

- If n is used, force the specified level number to n.
- If +n is used, add n to the current value of the specified level number.
- If -n is used, subtract n from the current value of the specified level number.

3.5.2.3 .STYLE HEADERS

This command causes a .BREAK and re-formats the header levels, and/or the case, as specified by three number-parameters (n1, n2, n3).

- The n1 parameter forces a run-in format, normally only used with .HL3 through .HL6, for all of the header levels that are equal to and greater than the specified value (i.e., if n1 = 4, use a a run-in format for all .HL4's, 5's, 6's, etc.); all lower numbered levels will then use a non-run-in format (i.e. .HL1 through .HL3).
- The n2 parameter capitalizes the text (text2) for all of the header levels that are equal to and less than the specified value. (e.g.,if n2 = 4, capitalize text2 for all .HL1's through .HL4's).
- The n3 parameter capitalizes the first letter of the text (text2) for all of the header levels that are equal to and less than the specified value (e.g., if n3 = 6, capitalize the first letter for all .HL1's through .HL6's).

The command may be formatted as follows:

.STYLE HEADERS n1, n2, n3

.STHL n1, n2, n3

Terminators: {.|EL|;|!}

Characteristics:

- If n1 is greater than the largest valid level (i.e., 6) all header levels will be non-run-in.

- If valid levels are not specified (i.e., less than 1 or greater than 6) for n2 or n3, the text is left in the input case (unless flags are used).

- If n2 and n3 specify the same or overlapping values, the n2 parameter (capitalize all words) takes precedence.

Defaults:

The default values for n1, n2, and n3, are 3 (run-in .HL3), and 6 (capitalize first letters for .HL2's through .HL6's).

3.5.2.4 .DISPLAY LEVELS

This command causes a .BREAK and transforms header level numbers into a format specified by the user (i.e., decimal, octal, hexadecimal, roman numerals, or letters). Moreover, the case (upper, lower, mixed) of all roman numerals or letters can be specified.

The command does not change values, but merely outputs equivalent values in the format specified.

Since each position of a legal header level (except the section prefix) can be independently formatted by separate parameters (y1 through y6), combinations of formats are possible.

For example, for CHAPTER 2, a .HL3 of 2.2.1.1 can be displayed as a 2.B.1.1 or a 2.2.1.a by coding y1 = LU or y3 = LL.

However, note that the section prefix (2 in the example) is evoked by a section command (i.e., .CHAPTER, .NUMBER CHAPTER, or .DISPLAY CHAPTER), is effectively header level zero, and cannot be affected by a .DISPLAY LEVELS parameter (y1 - y6).

The command uses up to six display-descriptor-parameters (refer to section 2.1.3.7 for codes).

The command may be formatted as follows:

```
.DISPLAY LEVELS y1, y2, y3, y4, y5, y6

.DHL y1, y2, y3, y4, y5, y6
```

Terminators: {.|EL|;|!}

Characteristics:

- Excluding the section prefix (S), the y1 through y6 parameters equate with the six legal header level positions (e.g., .HL6 = S.1.2.3.4.5.6).

- The .DISPLAY LEVELS command must be inserted before the .HEADER LEVEL commands the user desires to affect.

3.5.3 Appendices and Numbering

An appendix is a major section defined as an appendix by command. The identifying letter for an appendix (i.e., A, B, C, etc.), whether imposed by DSR or by the user, becomes the prefix for all of the header levels used within the appendix.

3.5.3.1 .APPENDIX

This command causes a .BREAK and initiates an appendix, taking the text-parameter (textl) directly following the keyword as the title. The identifying letter will be automatic and sequential, starting with Appendix A.

The command is formatted as follows:

```
.APPENDIX textl

.AX textl
```

Terminators: {.|EL|;|!}

Characteristics:

1. When the command is executed a new page is automatically invoked and 12 blank lines are inserted.

2. The left margin is set to zero, and fill, justify, and single-spacing are enabled.

3. In upper case, the word "APPENDIX" followed by a space and an appropriate letter (i.e., A, B, C, etc.) is centered on the thirteenth line.

4. One blank line is then inserted, followed by the centered title (textl). The title will be printed entirely in upper case, unless flags in the original text specify otherwise.

5. Three blank lines are then inserted; thus, the first line of the body of the text (i.e., header level one) will appear on the ninteenth line.

3.5.3.2 .NUMBER APPENDIX

This command is used to force an initial identifying appendix letter, following which the sequential lettering of subsequent new appendices will be resumed. The command also re-enables page numbering.

A count-parameter (c) is used with the command to specify the appendix letter(s) via letters or digits.

If the parameter consists of a letter or letters, the new appendix sequence starts with the letter(s) specified.

If the parameter consists of a digit or digits, the new appendix sequence starts with the letter equivalence of the numeric value of the digit (i.e., 1 through 26 = a through z), or digits (i.e., 27 through 52 = aa through az, etc.). Signed numbers may also be used to affect a relative change to the appendix letters by addition or subtraction.

The command is formatted as follows:

```
.NUMBER APPENDIX c

.NMAX c
```

Terminators: {.|EL|;|!}

Characteristics:

- If c consists of a letter, the appendix letter is changed to c.
- If c consists of a digit, the appendix letter is changed to the eqivalent letter value of c.
- If +c is used, add c to the current numeric value of the appendix letter(s).
- If -c is used, subtract c from the current numeric value of the appendix letter(s).
- If a .CHAPTER command directly follows the .NUMBER APPENDIX command, the word "CHAPTER" will be changed to "APPENDIX" and the specified letter(s) will be affixed; the reverse occurs if a .APPENDIX command directly follows a .NUMBER CHAPTER command.

3.5.3.3 .DISPLAY APPENDIX

This command causes a .BREAK and then transforms appendix numbering into a format specified by the user (i.e., decimal, hexadecimal, roman numerals, or letters). Moreover, the case (upper, lower, mixed) of all roman numerals or letters can be specified.

The command does not change values, but merely outputs equivalent values in the format specified.

To accomplish this, the command uses a display-descriptor-parameter code (refer to section 2.1.3.7 for codes).

The command may be formatted as follows:

```
.DISPLAY APPENDIX y

.DAX y
```

Terminators: {.|EL|;|!}

Characteristics:

The .DISPLAY APPENDIX command should be used at the top of a page, following the commands the user desires to affect (i.e., .APPENDIX and .NUMBER APPENDIX).

3.6 MISCELLANEOUS FORMATTING COMMANDS

3.6.1 .STANDARD

This command causes a .BREAK, re-enables the Paging Mode (i.e., .PAGING) (and other defaults), and sets certain page size parameters.

When the command is executed, the left margin is set to zero, the right margin to the current page-width, and fill, justify, and single-spacing are enabled.

An optional, signed (+/-) or unsigned, number-parameter (n) may be used to specify the page-width the user desires. However, for certain values of n (60 and 70) specific page-lengths (58 and 74) will be automatically evoked.

If a page-width of 60 or 70 is specified by n (or is a resultant value), the function of the command will be similar to that of a .PAGE SIZE command, allowing the number of lines per page to be automatically set to 58 or 74, respectively.

The command may be formatted as follows:

```
.STANDARD n

.SD n
```

Terminators: {.|EL|;|!}

Characteristics:

- If n is not used, the current page-width is used by default.
- If +n is used, n is added to the current page-width.
- If -n is used, n is subtracted from the current page-width.

Default:

The default value for n is the current page-width.

3.6.2 .COMMENT

This command allows the user to insert descriptive or explanatory comments (text2) in the input file that will not appear in the output file.

The command may be formatted as follows:

```
.COMMENT text2

.!  text2
```

Terminators: {EL|;)

Examples:

For several examples of the use of commentaries, refer to section 2.1.3.4 Exclamation Mark and 2.2.3.9 The COMMENT Flag.

3.6.3 .REQUIRE

This command allows the user to request that an externally located file, specified by a quoted-string parameter ("q") be processed as part of the input. This feature allows the user to piece together separately formatted input files to form a single output file.

Required files may be nested in the sense that an input file may require one or more files, while each required file may in turn require additional files. The sequence of these requests is, of course, implementation dependent.

The DSR program will recognize any DEC-standard file-specification. However, the file-specification must be enclosed in either single (') or double (") quotes.

For transportability purposes (i.e., multi-system usage), filenames should follow the form: "filnam.ext"; where filnam is no more than nine characters long and type (.ext) is no more than three.

Note that this command must be entered alone on a line; therefore, termination is by end-of-line code only.

The command may be formatted as follows:

```
.REQUIRE "q"

.REQ "q"
```

Terminators: {EL}

Example: .REQ "PLAN.RNO"

3.6.4 .CONTROL and .NO CONTROL CHARACTERS

These commands cause the recognition of control characters as normal text to be enabled or disabled.

The enabling command is inserted at the beginning of a file to allow non-printable characters and codes, that normally are unrecognizable to DSR, to be output as normal text.

If control characters exist in an input file and they are not collectively enabled an error message will occur when the output file is created.

The commands may be formatted as follows:

Enable	Disable
.CONTROL CHARACTERS	.NO CONTROL CHARACTERS
.CC	.NCC

Terminators: {.|EL|;|!}

3.6.5 .IF, .IFNOT, .ELSE, and .ENDIF

These commands are used to exert conditional control over the processing of segments of text and commands.

Each command requires physical column one placement; therefore, each command must be entered on a separate line.

The commands are entered as a group (e.g. .IF, .ELSE, .ENDIF), and each command uses the same name-parameter (name) to both define a segment and serve as the basis for a true or false test.

Test values are specified by the user via a command line switch (/VARIANT:name). The results of a test determine whether a segment will be processed by the DSR (true) or ignored (false).

An initiating command (.IF name or .IFNOT name) precedes a segment and specifies its test name. An .IF command is true if its specified name is contained in the switch variable (e.g., name1, name2, etc.). An .IFNOT command is true if its specified name is not contained in the switch variable.

An optional command (.ELSE name) is used to provide an alternative segment, and can only test true if its associated initiating command is false (i.e., .IF does not match or .IFNOT does). For example, if the segment specified by an .IF cannot be processed (no name match), the segment specified by the .ELSE will be processed.

A terminating command (.ENDIF name) is used to terminate the group of conditional commands.

Since a group of conditional commands controls a specific segment of text and commands, groups may be nested to expand selection possibilities. Thus, with the conditional command feature, selected information can be output from a multi-purpose input file.

Finally, a switch (/DEBUG) is also available which can force DSR to ignore the conditional commands, allowing all segmented information to be processed (refer to section 7.4.14).

The commands may be formatted as follows:

```
.IF name
.ELSE name
.ENDIF name

or:

.IFNOT name
.ELSE name
.ENDIF name
```

Terminators: {.|EL|;|!}

Example:

This example is in three parts:

The first part depicts the entry of three groups (A, B, C) of conditional commands, with groups B and C nested in group A.

The second part (Figure 3-4) depicts the logical flow of the entire structure and the selection possibilities.

The third part lists the selection possibilities and resultant output.

```
.NF
.IF   A                 !initiate Segment A
  A1 Segment            !process A1 if A is true.
.IF     B               !initiate Segment B.
  B1 Segment            !process B1 if A and B are true.
.ELSE   B               !alternative Segment B.
  B2 Segment            !process B2 if A is true and B is false.
.ENDIF  B               !end B group segment
  A2 Segment            !process A2 if A is true.
.ELSE A                 !alternative Segment A.
  A3 Segment            !process A3 if A is false.
.IF     C               !initiate Segment C.
  C1 Segment            !process C1 if A is false and C is true.
.ELSE   C               !alternative Subsegment C.
  C2 Segment            !process C2 if A and C are false.
.ENDIF  C               !end C group segment.
  A4 Segment            !process A4 if A is false.
.ENDIF A                !end A group segment.
```

Figure 3-4 depicts the logical flow:

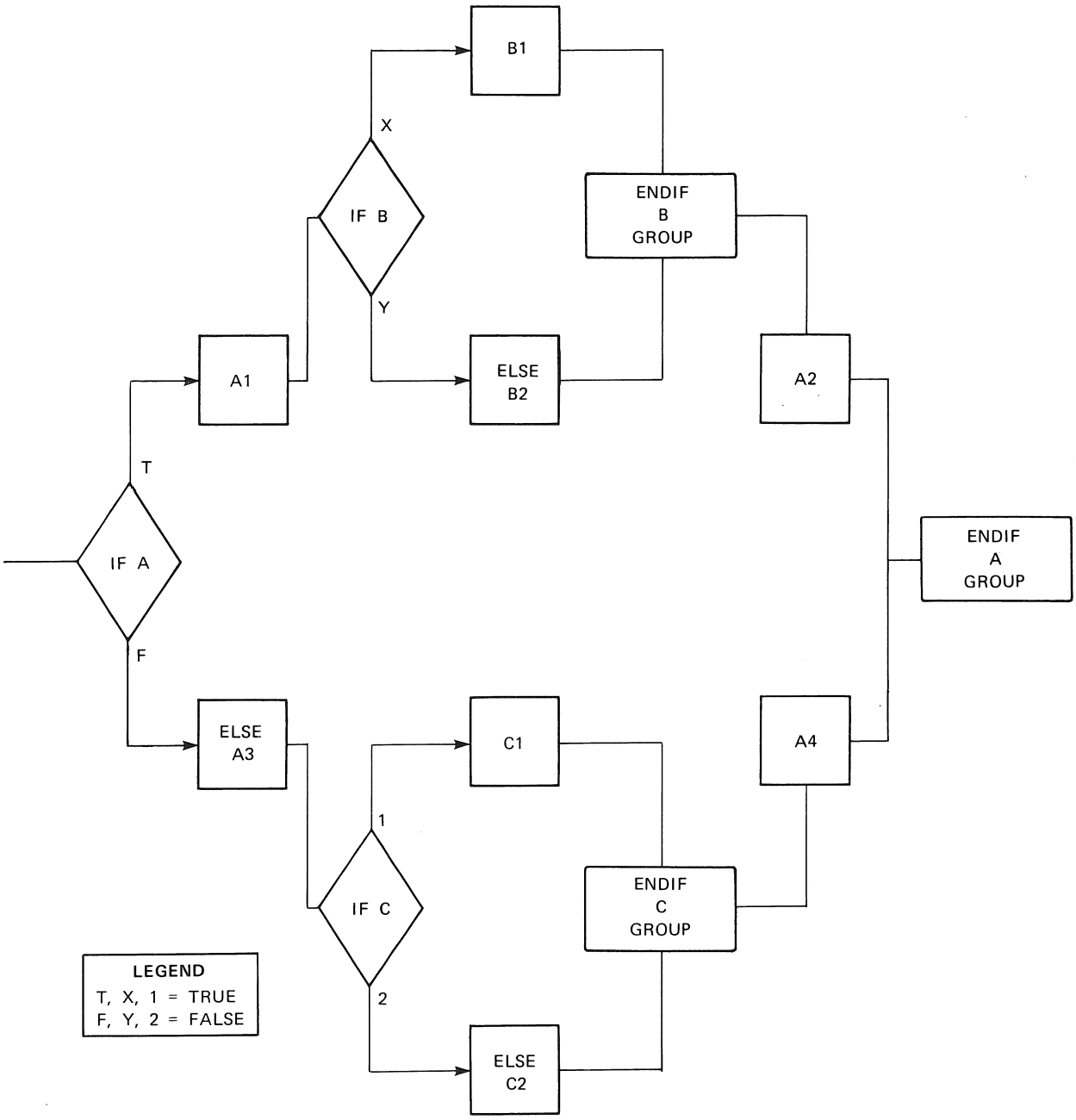


Figure 3-4 Nested Conditional Command Logic Flow

The following processing selections are possible:

- If no switch is used, the A3, C2, and A4 segments will be processed.

  A3 Segment

  C2 Segment

  A4 Segment

- /VARIANT:A will cause the A1, B2, and A2 segments to be processed.

  A1 Segment

  B2 Segment

  A2 Segment

- /VARIANT:B will cause the A3, C2, and A4 segments to be processed.

  A3 Segment

  C2 Segment

  A4 Segment

- /VARIANT:C will cause the A3, C1, and A4 segments to be processed.

  A3 Segment

  C1 Segment

  A4 Segment

- /VARIANT:"A,B" will cause the A1, B1, and A2 segments to be processed.

  A1 Segment

  B1 Segment

  A2 Segment

- /VARIANT:"A,C" will cause the same processing as /VARIANT:A.

- /VARIANT:"A,B,C" will cause the same processing as /VARIANT:A.

- /DEBUG will cause all segments to be processed.

A1 Segment

B1 Segment

B2 Segment

A2 Segment

A3 Segment

C1 Segment

C2 Segment

A4 Segment

3.6.6 .VARIABLE

The .VARIABLE command is used in conjunction with the /DEBUG switch to allow the user to not only output the text, defined by a group of conditional commands (3.6.5), but to label (flag) each segment as to its relative position (true or false path) in the IF-ELSE-ENDIF or IFNOT-ELSE-ENDIF structure.

To label the segments, the .VARIABLE command uses three parameters (name and 2 flags).

With the name-parameter, the segments defined by a group of conditional commands can be specified for evaluation. Consequently, with the flags-parameter pair, the named segments will be appropriately labeled in the output (e.g., a "T" flag for true and an "F" flag for false) to indicate the path in which the segment lies (e.g., the .IF true path or the .IF false path).

Since the /DEBUG switch will cause all text and commands to be processed, separate .VARIABLE commands must be entered for each conditional command group the user desires to examine.

In addition, separate pairs of flags (e.g., T,F; X,Y; 1,2; etc.) should be used to label each related segment. This prevents confusion when the user attempts to associate related segments with a specific group of commands.

The command may be formatted as follows:

.VARIABLE name flags

.VR name flags

Terminators: {.|EL|;|!}

Characteristics

- .VARIABLE commands must be inserted before any group of associated conditional commands are named and inserted.
- Separate .VARIABLE commands must be entered for every group of conditional commands for which labelling is desired.
- When using the /DEBUG switch, the /VARIANT switch has no effect.

Defaults:

If flags are not specified, the default labels will be spaces.

Example:

entering:

```
.VR A T,F              !T = true and F = false.
.VR B X,Y              !X = true and Y = false.
.VR C 1,2              !1 = true and 2 = false.
.NF
.IF A
   A1 Segment          !path if A is true.
.IF B
   B1 Segment          !path if A and B are true.
```

```
.ELSE B
   B2 Segment          !path if A is true and B is not.
.ENDIF B
   A2 Segment          !path if A is true.
.ELSE A
   A3 Segment          !path if A is false.
.IF C
   C1 Segment          !path if A is false and C is true.
.ELSE C
   C2 Segment          !path if A and C are false.
.ENDIF C
   A4 Segment          !path if A is false.
.ENDIF A
```

/DEBUG produces:

```
T   A1 Segment         !T = A1 is in .IF A true path.
X   B1 Segment         !X = B1 is in .IF B true path.
Y   B2 Segment         !Y = B2 is in .IF B false path.
T   A2 Segment         !T = A2 is in .IF A true path.
F   A3 Segment         !F = A3 is in .IF A false path.
1   C1 Segment         !1 = C1 is in .IF C true path.
2   C2 Segment         !2 = C2 is in .IF C false path.
F   A4 Segment         !F = A4 is in .IF A false path.
```

When the /DEBUG switch is used, all text is output and the specified structures may be analyzed (refer to flow diagram Figure 3-4).

3.6.7 .SET DATE and .SET TIME

These commands cause a .BREAK and respectively set a current, or specified, date (day, month, year) and time (hours, minutes, seconds).

By using signed (+/-) or unsigned number-parameters (n1, n2, n3) values may be adjusted or specified; otherwise current values are used.

Once the values are set, a SUBSTITUTE flag ($) may then be used to evoke any or all of the settings (e.g., $$date or $$time or $$year, etc.) as desired.

The commands may be formatted as follows:

```
.SET DATE n1, n2, n3

.SDT n1, n2, n3


.SET TIME n1, n2, n3

.STM n1, n2, n3
```

Terminators: {.|EL|;|!}

Characteristics:

- If no parameters are used, the current system date/time is evoked.
- The n1 parameter sets "day" (dd) or "hours" (hh).
- The n2 parameter sets "month" (mmm) or "minutes" (mm).
- The n3 parameter sets "year" (yy) or "seconds" (ss).
- If +n is used, n is added to the current value.
- If -n is used, n is subtracted from the current value.

Default:

If n is replaced with a comma, the corresponding value is unchanged.

Example 1:

entering:

```
.NO FILL
.SET DATE 12 5 63
.SET TIME 20 45 53
.FLAGS SUBSTITUTE
$$date
$$time
$$year
$$month
$$day
$$hours
$$minutes
$$seconds
$$day#$$month,#$$year
```

produces:

```
12 May 63                !dd, mmm, yy
20:45:53                 !hh:mm:ss
63                       !year
May                      !month
12                       !day
20                       !hours
45                       !minutes
53                       !seconds
12  May, 63              !day, month, year
```

Example 2:

entering:

```
.NO FILL
.SET DATE +1 -3 +2
.SET TIME -5,,22
.FLAGS SUBSTITUTE
$$date
$$time
$$year
$$month
$$day
$$hours
$$minutes
$$seconds
$$day#$$month,#$$year
```

produces:

```
13 Feb 65              !dd, mmm, yy
15:45:22               !hh:mm:ss
65                     !year
February               !month
13                     !day
15                     !hours
45                     !minutes
22                     !seconds
13  February, 65       !day, month, year
```

CHAPTER 4

DSR FLAGS AND FLAG CONTROL COMMANDS

4.1 THE DSR FLAGS

In Chapter 2, the basic rules governing flag usage were defined and the use of the default flags described.

In the following material, details of flag recognition and command control are provided along with descriptions of all of the flags available to the DSR (refer to Table 4-1).

In Table 4-1, all of the flag names are listed, along with the characters associated by default, and the general function of each flag is described.

Table 4-1

FLAG DESCRIPTIONS

FLAG NAME	CHAR	PURPOSE	NOTE
CONTROL	.	introduce DSR command	1,3
COMMENT	!	begin a comment	1,3
SUBSTITUTE	$	insert date or time	4
ACCEPT	_	take next character as text	1,2
UPPERCASE	^	upper case next character	1,3
LOWERCASE	\	lower case next character	1,3
CAPITALIZE	<	capitalize the next word	2
UNDERLINE	&	underline next character	1,4
BOLD	*	bold-face-type next char.	3
SPACE	#	force unexpandable space	1,2
SUBINDEX	>	subindex next word	1,2,5

INDEX	>	index the next word	2,5
HYPHENATE	=	hyphenate word	2
BREAK	|	break word between lines	2
PERIOD	+	double-space after char.	2
OVERSTRIKE	%	overstrike previous char.	2

NOTES

1. Character recognition is initially enabled.
2. Character is only used singly.
3. Character is used singly and as one of a pair.
4. Character is only used as one of a pair.
5. Refer to Chapter 5 (Creating an Index).

Notice in Table 4-1, that Note 3 specifies flags that can be paired to affect an operation, while Note 4 specifies a flag character that must be paired to be effective. These specifications exclude pairings with the ACCEPT flag, which can be paired with any character including itself.

4.1.1 Flag Character Recognition

With the exception of the COMMENT and CONTROL flags, a .FLAGS command can be formatted to collectively enable (.FLAGS ALL) or disable (.NO FLAGS ALL) the recognition of all of the flag characters.

Alternatively, every flag character can be individually enabled or disabled by using its flag name as a parameter (instead of ALL) with the .FLAGS command.

Thus, in regard to the state (enabled or disabled) of any given flag, the .FLAGS Command can be formatted to affect two separate conditions: one in relation to the flag considered collectively (noting exceptions), and the other in relation to the flag considered individually.

Because of these relationships, comparative enabled states must be achieved before any flag character can be recognized as being enabled by the program.

For example, if all flags are collectively disabled (.NO FLAGS ALL), the subsequent enabling of any individual flag (e.g., .FLAGS BOLD) will not evoke its recognition by the program.

Alternatively, if a flag is individually disabled (e.g.,.NO FLAGS BOLD), the subsequent enabling of all flags (.FLAGS ALL) will not evoke recognition of the disabled flag (BOLD) by the program.

Further, the user can, via the .FLAGS command, replace any flag character with another (i.e., another letter, number, or special character).

As noted (Note 1) in Table 4-1, nine flags are initially recognized by default while the eight remaining are not.

In regard to the eight flags initially disabled, or any flag disabled via command, without recognition such flags are inoperative and will, if used, be taken as normal text characters. Also, it is not possible to disable the recognition of the single occurrence of a flag character without also disabling its possible paired occurrence.

In summary: For a flag to be recognized it must be both collectively and individually enabled. Further, recognition only means that a flag character will not be taken as text, it does not mean that a flag's function can be necessarily performed. For example, in the case of the SUBINDEX flag, the functioning of the flag requires the execution of its associated DSR command (i.e., the .INDEX command).

4.1.2 Command and Case Flag Characters

4.1.2.1 The CONTROL Flag (.)

This flag is enabled by default, provides entry to the command recognition state, and may be paired as follows:

- Following its own occurrence in column one (..), where it will allow itself to be taken as a normal text character.

- Following an ACCEPT flag contained in column one (_.), which will also allow it to be recognized as a normal text character.

Note that when the CONTROL flag is defined as a normal text character by the ACCEPT flag, the default character (.) cannot be taken as end-of-sentence punctuation; therefore, a double-space will not follow the period (refer to section 3.3.5.6). This is useful if certain abbreviations (e.g., Mr. and Mrs.) are used within a sentence that would be misconstrued by the DSR as end-of-sentence punctuation.

4.1.2.2 The UPPERCASE Flag (^)

This flag is enabled by default and, as a single character, serves the same purpose as a typewriter shift key. The flag will capitalize any single letter that directly follows it, and has no effect if the following character is not a letter.

The UPPERCASE flag may be paired as follows:

- With a CAPITALIZE flag (^<), to initiate the capitalization of the text it precedes.
- With an UNDERLINE flag (^&), to initiate the underlining of the text it precedes (refer to sections 2.2 and 3.3.7.1).
- With a BOLD flag (^*), to initiate the bolding of the text it precedes (refer to section 3.3.7.3).
- With itself (^^) to affect a case-lock (i.e., no case change).

Setting a case-lock with the flag pair on a fixed-case terminal will invoke an upper case mode, while setting a case-lock on the variable-case terminal will invoke a lower case mode. This occurs because the normal mode of operation for each type of terminal will be imposed by the no-case-change (refer to 4.1.7 Fixed-Case Terminals).

4.1.2.3 The LOWERCASE Flag (\)

This flag is enabled by default, and causes the letter that directly follows it to appear in lower case. The flag has no effect if the following character is not a letter.

The LOWERCASE flag may be paired as follows:

- With the UNDERLINE Flag (\&), to terminate the underlining of the text it precedes (refer to sections 2.2 and 3.3.7.1).
- With the BOLD Flag (*), to terminate the bolding of the text it precedes (refer to section 3.3.7.3).
- With itself (\\), to affect a case-lock (i.e., force lower case) on all subsequent characters (refer to section 4.1.7 Fixed-Case Terminals).

4.1.3 Enhancement Flag Characters

4.1.3.1 The ACCEPT Flag (_)

This flag is enabled by default and causes any character that directly follows it to be accepted as normal text.

When the flag is used, the affected character will be taken as is and no spacing (other than that inserted by the user) will be added.

The latter effect (no automatic spaces) is significant when the user enters terms like "Mr." or "Mrs.", in which a period must be followed by a single space. By using the flag before the period (i.e., Mr_.) the character is not mis-construed by DSR as end-of-sentence punctuation and a double-space is not inserted by default (refer to section 3.3.5.6).

For underlining purposes, the flag can be used to force the acceptance of a real space (refer to section 2.2).

4.1.3.2 The SPACE Flag (#)

This flag is enabled by default, and allows an unexpandable space (not affected by justification) to appear in the output for every flag character inserted in the input file.

The flag may directly follow an UNDERLINE flag (&#) to affect the underlining of an unexpandable space (refer to sections 2.2 and 3.3.7.1).

4.1.3.3 The UNDERLINE Flag (&)

This flag is enabled by default, and its single occurence allows the next character to be underlined.

The operation performed by this flag can be disabled and resumed by command (refer to section 3.3.7.1).

The UNDERLINE flag may be paired as follows:

- With the UPPERCASE flag (^&) to lock underling, and with the LOWERCASE flag (\&) to unlock underlining (refer to section 2.2).

- With the SPACE Flag (&#), to affect the underlining of unexpandble spaces (refer to section 2.2).

Subsection 2.2 provides several examples of the use of the underlining feature. However, header level numbers and associated text can also be underlined by placing the initiation (^&) and termination (\&) pair, respectively, on the line before and the line after the .HEADER LEVEL command:

entering:

```
^&
.HL3;Title text is here.
\&
```

produces:

3.2.1.1 Title text is here.

4.1.3.4 The BOLD Flag (*)

The recognition of this flag is initially disabled. Therefore, the flag must be enabled by the insertion of a .FLAGS BOLD command.

If enabled (collectively and individually), the single character occurrence of the flag allows the next character to be bolded (i.e., overstruck once), for example:

entering:
```
.FLAGS BOLD
Use Types *A and *C
```

produces: Use Types A and C

In addition, the operation performed by this flag can be independently enabled or disabled by command (refer to section 3.3.7.3).

The BOLD Flag may be paired as follows:

With the UPPERCASE Flag (^*), to lock bolding until termination is effected, and with the LOWERCASE Flag (*), to unlock bolding. In this mode, all characters within the range of the initiation (^*) and termination (*) pair, will be bolded.

Entering: ^*These words are bolded.*

produces: **These words are bolded.**

4.1.3.5 The OVERSTRIKE Flag (%)

The recognition of this flag is initially disabled. Therefore, the flag must be enabled by the insertion of a .FLAGS OVERSTRIKE command.

If enabled (collectively and individually), and inserted between two characters, the flag allows the preceding character to be overstruck by the character following the flag.

This allows characters to be produced that are not normally available; such as, an "O" overstruck with a slash (Ø):

entering: .FLAGS OVERSTRIKE

Overstrike this O%/.

produces: Overstrike this Ø.

In addition, the operation performed by this flag can be independently enabled and disabled by command (refer to section 3.3.7.4).

4.1.3.6 The HYPHENATE Flag (=)

The recognition of this flag is initially disabled. Therefore, the flag must be enabled by the insertion of a .FLAGS HYPHENATE command.

If enabled (collectively and individually) and inserted between one or more syllables of a word, DSR will determine if and where the word should be broken and a hyphen inserted.

However, if DSR does not find it necessary to break the word the hyphen will not appear.

In addition, the operation of this flag can be independently enabled or disabled by command (refer to section 3.3.7.2).

entering:

```
.F.RM30
.FLAGS HYPHENATE
.br;This is an example of a hyph=en=at=ed word.
```

produces:

This is an example of a hyph-
enated word.

4.1.3.7 The BREAK Flag (|)

The recognition of this flag is initially disabled. Therefore the flag must be enabled by the insertion of a .FLAGS BREAK command.

The flag is used to indicate the logical break(s) in a sequence of related characters, such as a mathematical formula or a hyphenated phrase.

If enabled (collectively and individually), and inserted at logical break points, DSR will only break the sequence as directed.

Example:

entering:

```
.F.RM40
.FLAGS BREAK
.b;This is an example of a phrase (end-|of-|line) with
logical break points.
```

produces:

This is an example of a phrase (end-of-
line) with logical break points.

4.1.3.8 The PERIOD Flag (+)

The recognition of this flag is initially disabled. Therefore, the flag must be enabled by the insertion of a .FLAGS PERIOD command.

The flag is used to allow a double-space to be inserted directly following any character. For instance, the flag may be used to insert a double-space following end-of-sentence punctuation that is unconventional (i.e., not a period, semi-colon, colon, question mark, or exclamation mark).

If enabled (collectively and individually), and in .FILL mode, a double-space will occur when the flag is inserted directly following the end-of-sentence character. However, a normal end-of-sentence space must be inserted for the PERIOD flag to be effective.

For example, the user may want to end a sentence with a another complete sentence that is enclosed in either quotes or parentheses:

entering:

```
.FLAGS PERIOD

.br;The answer was not clear.  "What do you  mean?"+  There
was no response.
```

produces:

The answer was not clear. "What do you mean?" There was no response.

4.1.3.9 The CAPITALIZE Flag (<)

The recognition of this flag is initially disabled. Therefore, the flag must be enabled by the insertion of a .FLAGS CAPITALIZE command.

If enabled (collectively and individually), the CAPITALIZE flag causes all the letters in the word directly following it to be capitalized, with the exception of any letters that may be flagged by an ACCEPT (_), UPPERCASE (^), or LOWERCASE (\) flag.

With the latter constraints, capitalization will continue until:

- a real or flagged space occurs.
- a BREAK flag (|) occurs.
- a HYPHENATE flag (=) occurs.

- an end-of-line code occurs.

- another CAPITALIZE flag occurs.

Note that another CAPITALIZE flag affects a return to the standard (upper or lower) case mode (refer to section 4.1.7 Fixed-Case Terminals).

4.1.4 Miscellaneous Flag Characters

4.1.4.1 The COMMENT Flag (!)

This flag is enabled by default and can be used in place of the .COMMENT Command to allow the insertion of user comments in and for the .RNO file (i.e., the commentary will not appear in the output file).

When the flag is inserted in a multi-command string (e.g., .LM0!) it becomes both the terminator of the previous command and the initiator of the commentary.

However, if the flag is to be inserted at the begining of a line it must be preceded by a column one CONTROL flag (e.g., .!); thus, in the latter case, it will be entered as one of a flag pair (refer to sections 2.2 and 3.6.2).

4.1.4.2 The SUBSTITUTE Flag ($)

The recognition of the flag is initially disabled. Therefore, the flag must be enabled by the insertion of a .FLAGS SUBSTITUTE command. In addition, this is the only flag that requires the insertion of a pair of characters ($$) to be effective.

If enabled (collectively and individually), the flag allows either the date or time to be output, as defined by a name-parameter associated with the formatting of the flag pair.

The following example demonstrates the use of the flag and will always reflect the date and time when this document was generated:

entering:

```
.FLAGS SUBSTITUTE
$$Date
$$Time
$$Year
$$Month
$$Day
$$Hours
$$Minutes
$$Seconds
$$Day#$$Month,#$$Year
```

produces:

```
03 Oct 79
13:27:58
1979
October
3
13
27
58
3 October, 1979
```

4.1.5 Variable-Case Terminals

In the previous material the use of the flags has been described in relation to their use on lower-case terminals (i.e., variable-case).

Since several flags (as well as commands) are primarily designed for use with upper-case-only terminals, that also perform functions for lower-case terminals, differences in functionality may occur depending on the function performed and the type of terminal used. For this reason, the following information describes the differences encountered when an upper-case-only terminal (i.e., fixed-case) is used.

4.1.6 Fixed-Case Terminals

Fixed-case terminals operate in an upper-case mode only. This means that all input, echoed on the terminal, and all output will normally appear in upper case. However, by using three flag characters (UPPERCASE, LOWERCASE, and CAPITALIZE), variable-case output may be generated, as follows:

Case-Shifting a Fixed Terminal

Since a fixed-case terminal inputs in upper case only, the easiest method for implementing case-shifting is to initially force a lower-case-lock. This can be accomplished in one of two ways: by inserting either a .LOWER CASE command or a LOWERCASE flag pair (\\).

With the lower-case-lock established, any letter requiring capitalization can be preceded by a single UPPERCASE flag (^), while any group of letters requiring capitalization can be preceded by the UPPERCASE flag pair (^^) to invoke an upper-case-lock; this, in effect, simulates the setting of the shift lock on a typewriter. However, the latter can be temporarily accomplished by using the CAPITALIZE Flag (<), which only capitalizes a word and then returns to the current case mode (regardless of the type of terminal used).

Thus, the CAPITALIZE Flag operates as if two circumflexes (^^) had been inserted before the word and two backslashes (\\) inserted after the word; moreover, the use of this method ensures a return to the lower-case-lock mode.

At this point it is important to note that, like the UPPERCASE Flag pair (^^), when the .UPPER CASE command is used with a variable-case terminal, the no-case-change evokes a lower-case-lock (a decided misnomer in regard to the command name). However, when used with a fixed-case terminal, the no-case-change properly invokes an upper-case-lock (i.e., the normal mode of the device).

The following example depicts the use of the flag characters as they would appear first as input echoed on a fixed-case terminal and second as formatted text in the output file.

The input file would appear as follows:

```
\\^THIS SENTENCE HAS BEEN ENTERED VIA <DEC<SYSTEM-10
USING THE ^^UPPERCASE, LOWERCASE, \\AND ^^CAPITALIZE
\\FLAGS AS AN EXAMPLE.
```

The output file would appear as follows:

```
This sentence has been entered via DECsystem-10
using the UPPERCASE, LOWERCASE, and CAPITALIZE
flags as an example.
```

4.2 FLAG CONTROL COMMANDS

The recognition of all flag characters is controlled by two commands (.FLAGS and .NO FLAGS).

By using a flag name as a parameter a flag may be individually enabled (.FLAGS name) or disabled (.NO FLAGS name).

By excluding the flag name or using the word "ALL", all flags (except CONTROL and COMMENT) may be collectively enabled (.FLAGS ALL) or disabled (.NO FLAGS ALL).

Finally, any of the four terminator characters (./ EL/ ;/ !) can be used to terminate a flag control command.

4.2.1 Redefining a Flag Character

For special purposes, a flag character may be redefined by using a character-parameter (k) along with a .FLAGS command (refer to section 2.1.2.3 Character-Parameter).

Note, however, that the redefinition of a flag will affect any paired condition of the character.

An example follows of the redefinition of the BOLD flag character (by default an asterisk) to an ampersand:

```
.NO FLAGS UNDERLINE
.FLAGS BOLD &
```

Note that .NO FLAGS UNDERLINE is first inserted to disable the use of the ampersand character (available to underline by default) for underlining. If this is not done, bolding cannot assume the use of the character. This is due to the DSR sequence of default flag character recognition, in which the recognition of the underline character (&) precedes the recognition of the bold character (*); the sequence is shown in Table 4-1.

Dare-devil users may also replace a flag character with an ASCII control character. If this is attempted, the selected name of the code must be flagged for acceptance (via an ACCEPT flag) in the command line or it will not be recognized.

For example, the following will redefine the OVERSTRIKE flag (by default a percent sign) with a back-space control character (Control-H):

```
.FLAGS OVERSTRIKE _^H
```

4.2.2 All Flags and CONTROL Flag Control

4.2.2.1 .FLAGS ALL and .NO FLAGS ALL

These commands enable and disable the recognition of all existing flag characters, except CONTROL and COMMENT.

The commands may be formated as follows:

Enable	Disable
.FLAGS ALL	.NO FLAGS ALL
.FLAGS	.NO FLAGS
.FL	.NFL

4.2.2.2 .NO FLAGS CONTROL and .FLAGS CONTROL

Since there is no possible way to re-enable recognition of a CONTROL flag character (via .FLAGS CONTROL) once .NO FLAGS CONTROL is entered, these commands are mainly for compatability purposes. However, as with all such commands, .FLAGS CONTROL may be used to re-define the CONTROL flag (.).

The commands may be formatted as follows:

Disable	Resume
.NO FLAGS CONTROL	.FLAGS CONTROL
.NFL CONTROL	.FL CONTROL

Default:

CONTROL flag character (.) recognition is enabled by default.

4.2.3 Case Flags Control

4.2.3.1 .NO FLAGS UPPERCASE and .FLAGS UPPERCASE

These commands disable and re-enable the recognition of the UPPERCASE flag.

The commands may be formatted as follows:

Disable	Resume
.NO FLAGS UPPERCASE	.FLAGS UPPERCASE
.NFL UPPERCASE	.FL UPPERCASE

Default:

UPPERCASE flag character (^) recognition is enabled by default.

4.2.3.2 .NO FLAGS LOWERCASE and .FLAGS LOWERCASE

These commands disable and re-enable the recognition of the LOWERCASE flag.

The commands may be formatted as follows:

Disable	Resume
.NO FLAGS LOWERCASE	.FLAGS LOWERCASE
.NFL LOWERCASE	.FL LOWERCASE

Default:

LOWERCASE flag character (\) recognition is enabled by default.

4.2.4 Enhancement Flags Control

4.2.4.1 .NO FLAGS ACCEPT and .FLAGS ACCEPT

These commands disable and re-enable the recognition of the ACCEPT flag.

The commands may be formatted as follows:

Disable	Resume
.NO FLAGS ACCEPT	.FLAGS ACCEPT
.NFL ACCEPT	.FL ACCEPT

Default:

ACCEPT flag character (_) recognition is enabled by default.

4.2.4.2 .NO FLAGS SPACE and .FLAGS SPACE

These commands disable and re-enable the recognition of the SPACE flag.

The commands may be formatted as follows:

Disable	Resume
.NO FLAGS SPACE	.FLAGS SPACE
.NFL SPACE	.FL SPACE

Default:

SPACE flag character (#) recognition is enabled by default.

4.2.4.3 .NO FLAGS UNDERLINE and .FLAGS UNDERLINE

These commands disable and re-enable the recognition of the UNDERLINE flag.

The commands may be formatted as follows:

Disable	Resume
.NO FLAGS UNDERLINE	.FLAGS UNDERLINE
.NFL UNDERLINE	.FL UNDERLINE

Default:

UNDERLINE flag character (&) recognition is enabled by default.

4.2.4.4 .FLAGS BOLD and .NO FLAGS BOLD

These commands enable and disable the recognition of the BOLD flag.

The commands may be formatted as follows:

Enable	Disable
.FLAGS BOLD	.NO FLAGS BOLD
.FL BOLD	.NFL BOLD

Default:

BOLD flag character (*) recognition is enabled by default.

4.2.4.5 .FLAGS OVERSTRIKE and .NO FLAGS OVERSTRIKE

These commands enable and disable the recognition of the OVERSTRIKE flag.

The commands may be formatted as follows:

Enable	Disable
.FLAGS OVERSTRIKE	.NO FLAGS OVERSTRIKE
.FL OVERSTRIKE	.NFL OVERSTRIKE

Default:

OVERSTRIKE flag character (%) recognition is enabled by default.

4.2.4.6 .FLAGS HYPHENATE and .NO FLAGS HYPHENATE

These commands enable and disable the recognition of the HYPHENATE flag.

The commands may be formatted as follows:

Enable	Disable
.FLAGS HYPHENATE	.NO FLAGS HYPHENATE
.FL HYPHENATE	.NFL HYPHENATE

Default:

HYPHENATE flag character (=) recognition is enabled by default.

4.2.4.7 .FLAGS CAPITALIZE and .NO FLAGS CAPITALIZE

These commands enable and disable the recognition of the CAPITALIZE flag.

The commands may be formatted as follows:

Enable	Disable
.FLAGS CAPITALIZE	.NO FLAGS CAPITALIZE
.FL CAPITALIZE	.NFL CAPITALIZE

Default:

CAPITALIZE flag character (<) recognition is enabled by default.

4.2.4.8 .FLAGS SUBSTITUTE and .NO FLAGS SUBSTITUTE

These commands enable and disable the recognition of the SUBSTITUTE flag.

The commands may be formatted as follows:

Enable	Disable
.FLAGS SUBSTITUTE	.NO FLAGS SUBSTITUTE
.FL SUBSTITUTE	.NFL SUBSTITUTE

Default:

SUBSTITUTE flag character ($) recognition is enabled by default.

CHAPTER 5

CREATING AN INDEX

5.1 INTRODUCTION

During the editing of an input file, DSR can accommodate the simultaneous construction of an index.

To facilitate construction, an index buffer is provided into which specified items are stored prior to output.

To provide input to the buffer, DSR commands and/or flags are used to specify entry items.

Entries may be terms that are user devised and especially inserted in the file for the index, or terms that are directly extracted from the running text, including titles and header level information.

To output the buffer, a DSR command is inserted to specify at what point in the output file the index listing will be generated (generally the end of the file) and the format in which it will appear. Thus, when the input file is processed, the index will be output at a point in the document, and in a form, determined by the user.

An index can be constructed as a section, that is subsectioned in the same manner as a chapter or appendix (i.e., with header levels). But the simplest form merely consists of listed items, derived from the text, that are referenced to their point(s) of occurrence by page number.

The following describes the operation and use of all of the indexing facilities available with DSR.

5.1.1 Specifying Index Items

There are two ways of independently specifying items for an index: the storage command method and the INDEX flag method. Using these two methods, respectively, two types of entry can be specified:

- Via the command method, a multi-word-item is reserved for the index when it is specified as one of the storage command (.INDEX and .ENTRY) parameters.

- Via the INDEX flag method, a single-word-item is reserved for the index when it is flagged within the text.

For both methods: items are reserved for buffer storage by command, and by a commonly used flag character (>) which serves as a SUBINDEX flag for the command method and an INDEX flag for the INDEX flag method. (By default, the character is initially only available to the command method as a SUBINDEX flag).

5.1.1.1 The Storage Command Method

When the storage command method is used, a SUBINDEX flag is inserted before an item (parameter) to specify its level of entry in the buffer.

Thus, depending on how parameters are flagged within a given command, the index may be subdivided into levels of major and minor items (i.e., a major item followed by indented minor items).

For example:

```
Major item1
  minor item1
  minor item2
    minor item2a
    minor item2b
  minor item3
Major item2
```

The storage command method allows for the independent use of as many commands and parameters as desired to reference the subject matter.

The commands are inserted singly or as a group, on lines directly following the subject matter, and parameters (items) are selected which best reference the key ideas presented by the material. Therefore, items need not be duplicates of words from the subject matter, but may be descriptive phrases that the user feels are more suitable for an index. This is possible because the page numbers, appearing as references in the index, are not specified by the words of the subject matter, but rather by the pages on which the associated commands are located.

Note that the only functional difference between the two storage commands (.INDEX, and .ENTRY) is that .ENTRY does not provide reference numbers in the index.

5.1.1.2 The INDEX Flag Method

When the INDEX flag method is used, an item is flagged within running text, and only one level of entry (i.e., major item) is possible.

The INDEX flag is not available by default. To use this method, the INDEX flag must be enabled by the insertion of a .FLAGS INDEX command.

5.1.2 Specifying Index Output

There are two mutually exclusive ways of outputting an index. These are the output-index command method and the TCX program method:

- When an output-index command (.PRINT INDEX or .DO INDEX) is inserted at the bottom of the .RNO file, a one-column-index is generated.

- When the /INDEX command line switch is used during DSR processing, and in conjunction with the TCX utility program, a two-column-index is generated.

5.1.2.1 The Output Command Method

If the output command method is used, the one-column-index appears at the point of command insertion (generally the end of the document).

The major items are positioned at the left margin, and each item is directly followed by a series of dots, ending with the reference page number(s).

Note that this method is only supported by TOPS-10, TOPS-20, and VAX/VMS versions of DSR.

5.1.2.2 The TCX Program Method

If the TCX program method is used, the two-column-index is generated as a separate .RNO type file, that is defined as .RNX by the program.

The .RNX file is a standard DSR input file that can be:

- included with the main document (.RNO) via a .REQUIRE command.

- processed separately by DSR to produce an independent index file (.MEX).

When the .RNX file is processed, the entire index is split into two columns per page, with each column appearing in a format similar to a one-column index; however, no dots will appear between the items and page numbers.

An important benefit of the TCX program is the ability to create a master index file from separate intermediate index files (.BIX), that have, in turn, been created from existing .RNO files. When the separate intermediate files are merged (e.g., via a TOPS-20 APPEND command) and processed by TCX, the master index (.RNX) will be created.

Note that the TCX program method is supported by all versions of DSR.

5.2 INPUTTING THE INDEX

The following initially describes the differences that exist between the flags (SUBINDEX and INDEX) that are respectively used to specify items for the storage command and INDEX flag methods. The material then provides separate descriptions and examples of the use of each method.

Note that both methods may be used individually or simultaneously to create an index.

Flag Characteristics

Both flags use the same character (>); however, the SUBINDEX flag is enabled by default and the INDEX flag is not. Thus, the flag character is only initially available for use within the storage command structures as a SUBINDEX flag. If an attempt is made to use the character within the general text, without first enabling the INDEX flag, the specified item will not be collected and the flag character (>) will appear in the output.

The flags are reserved for their separate functions because:

- The SUBINDEX flag can collect all information up to a semi-colon, end-of-line code, or the next SUBINDEX flag; therefore, the flag can specify multi-word-items.

- The INDEX flag can only collect text information up to a space, SPACE flag (#), HYPHENATE flag (=), BREAK flag (|), EL code, or the next INDEX flag; therefore, the flag can only specify single-word-items.

Further, note that when the flag character is encountered within a storage command it can never be recognized as an INDEX flag character even if the INDEX flag is enabled.

5.2.1 Using the Storage Commands

The following describes the manner in which items are specified for an index by both of the storage commands (.INDEX, and .ENTRY).

The general form of the commands is as follows:

```
.Keyword item0>item1>item2....>itemn
```

5.2.1.1 The SUBINDEX Flag (>)

Although this flag is initially enabled by default, it is only functionally operative when it is used within an .INDEX or .ENTRY command parameter.

The SUBINDEX flag causes all the parameter characters that directly follow the flag to be included in the index until the end of a parameter is indicated by the appearance of:

- a semi-colon.
- an end-of-line code.
- another SUBINDEX flag (>).

Unlike the INDEX flag, a SUBINDEX flag cannot be independently used throughout the text to collect information for the index.

5.2.1.2 .INDEX Command

This command causes the text-parameter (text2), directly following the keyword, to be delivered to the index buffer.

If the command is used without a SUBINDEX flag (>), text2 is delivered to the buffer as a major item (Item0), which will be referenced by its page number, for example:

entering:

```
.INDEX item0
```

produces:

```
Item0. . . . . .page no.
```

If the command is used with one or more SUBINDEX flags, the command will subdivide the entry into a major item and one or more minor items (Item0, item1, item2, etc.), all of which will be inserted (and properly indented) in the buffer. However, only the minor items are preceded by a flag, and only the last item (itemn) will be referenced by page number, for example:

entering:

```
.INDEX item0>item1>item2
```

produces:

Item0

item1

item2. . . . . .page no.

Therefore, to provide reference numbers for all of the items within a subdivided group, the group must be gradually assembled in the buffer via separate commands. This means that current major and minor items already in the buffer must be identically repeated with each new command, along with the flags previously used (i.e., any typos, case or flag differences will result in separate instead of group entries). With this arrangement, levels of insertion can be identified by DSR and each new entry (itemn) can be inserted at the proper level.

The concepts just described are illustrated in the following example, where three commands are separately used to assemble a group of related items in the buffer.

The resultant content of the buffer is shown at each stage merely to depict the manner in which an appropriate reference-and-level is achieved for the last item in each command (itemn). In this regard, note that in the first stage Item0 = itemn, in the second item1 = itemn, and in the third item2 = itemn.

entering:

.INDEX item0

produces:

Item0. . . . . .page no.

entering:

.INDEX item0>item1

produces:

Item0. . . . . .page no.

item1. . . . . .page no.

entering

.INDEX item0>item1>item2

produces:

Item0. . . . . .page no.

item1. . . . . .page no.

item2. . . . . .page no.

Understand that if only the last command (containing all three items) had been used, without the two previous commands, the items would have been entered as a group but only item2 could evoke a page number.

Note, also, that the first letter of the major item (Item0) is printed in upper case, while all other item letters are in lower case. This is the manner in which the items will normally be printed (by default). However, the user can alter the case rules by the use of appropriate flags.

The commands may be formatted as follows:

```
.INDEX text2

.X text2
```

Terminators: {EL|;}

Characteristics:

- When the command is inserted, proper spacing must be maintained between words. This is important because input spacing (normal or flagged) is duplicated in the output.
- Index items may be enhanced by capitalization (complete or partial), overstriking, underlining, and/or bolding.
- As with conventional flag usage, no spacing is allowed between any inserted flags and the affected item.

Defaults:

The SUBINDEX flag is initially available by default, the INDEX flag is not.

.INDEX Examples

The following depicts the indexing of items using the .INDEX command, and the generation of the items as a one-column-index (refer to 5.3.1).

Example 1:

Inputting the following commands on page 5:

```
.index section formats
.index title formats
.index subject formats
.index section formats >section
.index section formats >chapter
.index section formats >appendix
.index section formats >index
.index title formats >first title
.index title formats >title
.index title formats >subtitle
.index subject formats >case
.index subject formats >margin
.do index
```

produces:

INDEX

Note that when the print-out is initiated via a .DO INDEX, a new page is forced and the word "INDEX" appears, centered and followed by four blank lines. The entries are against the left margin and listed in alphabetical order for each level. In addition, a blank line occurs between the major items having a different first letter, otherwise regular line spacing is used (refer to 5.3.1.2).

Example 2:

Inputting the following commands on page 15:

```
.flags capitalize
.entry <section <formats
.entry <subject <formats
.index <section <formats >section
.index <section <formats >chapter
.index <section <formats >appendix
.index <section <formats >index
.index <subject <formats >case
.index <subject <formats >case >upper
.index <subject <formats >case >lower
.index <subject <formats >margin
.index <subject <formats >margin >left
.index <subject <formats >margin >right
.print index
```

produces:

```
SECTION FORMATS
  appendix . . . . . 15
  chapter  . . . . . 15
  index  . . . . . 15
  section  . . . . . 15
SUBJECT FORMATS
  case . . . . . 15
    lower  . . . . . 15
    upper  . . . . . 15
  margin . . . . . 15
    left . . . . . 15
    right  . . . . . 15
```

Note that when the print-out is initiated via a .PRINT INDEX, a new page is not forced and the word "INDEX" is not printed; therefore, four blank lines do not occur (refer to 5.3.1.1). However, while the margins and the use of an alphabetic listing remain the same, a CAPITALIZE flag (<) has been used in the example to upper case the major items. Moreover, since the major items have been indexed by an .ENTRY command no reference numbers appear.

Example 3:

Inputting the following commands on page 3:

```
.flags capitalize
.entry \\^command <form<ats
.entry <command <formats
.index \\^command <form<ats ><section
.index <command <formats ><chap<ter
.index <command <formats ><^&app<endix\&
.print index
```

produces:

COMMAND FORMATS
APPendix 3
CHAPter 3
Command FORMats
SECTION 3

This example demonstrates how various flags (i.e., lower case, capitalize, underline) may be used to provide mixed case and underlined items.

Notice that the two major items (Command Formats), though identically spelled, are not identically formatted. Moreover, the items are separately printed in a specific order. This occurs in accordance with rules for printing identically spelled items, with the output order depending on the nature of the text enhancements (refer to subsection 5.3.3).

5.2.1.3 .ENTRY Command

This command is used in the same manner as the .INDEX command and provides the same output. However, the item appears alone; no dotted line or reference page number is included.

The command may be formatted as follows:

```
.ENTRY text2

.Y text2
```

Terminators: {EL|;}

Characteristics:

Since reference page numbers are not evoked by this command, it may be used to provide isolated internal headings (i.e., "blind entries") for the listing or to define minor items that may be found on a common page (the page being defined by the subentry items).

Defaults:

The SUBINDEX flag is available with this command by default.

.ENTRY Command Example

The following depicts the indexing of items via the .ENTRY command, and the generation of the items as a one-column-index (refer to 5.3.1).

Inputting the following commands on page 10:

```
.flags capitalize
.entry <console
.index ><console ><s<witches
.entry ><console ><s<witches ><add<ress mode switch
.entry ><console ><s<witches ><clr <err<ors switch
.entry ><console ><s<witches ><start switch
.entry ><console ><s<witches ><stop switch
.print index
```

produces:

```
CONSOLE
  Switches . . . . . 10
    ADDress mode switch
    CLR ERRors switch
    START switch
    STOP switch
```

Note that the page number (10) in the example is attached only to the subindexed item "switches".

5.2.2 Using the Index Flag Method

The following examples depict the manner in which INDEX flags (>) are used to specify items for the index.

5.2.2.1 The INDEX Flag (>)

The recognition of this flag is initially disabled. Therefore, the flag must be enabled for use by the insertion of a .FLAGS INDEX command.

The INDEX flag allows all the text characters that directly follow the flag to be included in the index until the end of text is indicated by the appearance of:

- a space (depression of SPACE bar or TAB key).

- an end-of-line code.
- a SPACE Flag (#).
- a BREAK flag (|).
- a HYPHENATE flag (=)
- another INDEX flag (>).

Thus, unlike the SUBINDEX flag, which is initially enabled and must be used within the range of SUBINDEX or INDEX command parameters, the INDEX flag can be used throughout the entire text to specify information (letters, characters, or words) for inclusion in the index.

5.2.2.2 .FLAGS and .NO FLAGS INDEX Commands

These commands enable and disable recognition of the INDEX flag. However, the .FLAGS INDEX command can also be used to redefine the INDEX flag character (refer to section 4.2 Flag Control Commands).

The commands may be formatted as follows:

Enable	Disable
.FLAGS INDEX	.NO FLAGS INDEX
.FL INDEX	.NFL INDEX

Default:

The INDEX flag character is a right angle bracket (>) by default.

INDEX Flag Examples

For the examples that follow, assume that a .FLAGS INDEX command is inserted at the top of the document and a .PRINT INDEX command at the bottom (refer to 5.3.3).

Example 1:

In this example, a header level is flagged for inclusion in the index and ACCEPT flags are used to force certain characters (DEC) to be taken literally:

entering:

```
.HL1 THE >_D_E_Ctape

.HL1 THE >MAGtape
```

produces:

```
DECtape . . . . . 10

Magtape . . . . . 20
```

Note in the last example, that only the first letter of the MAGtape entry is indexed in upper case, because an ACCEPT flag was not used.

Example 2:

In this example, spaces between words are preceded by ACCEPT flags to force a 3-word entry into the index:

entering:

```
>force_ these_ words into the index
```

produces

```
Force these words. . . . . 15
```

Example 3:

In this example, partially and completely underlined entries are formatted via the UNDERLINE flag (&) as follows:

entering:

```
The >&_D&_E&_Csystem-10 can be used.
The parameter requires a >^&value=15\&.
```

produces:

```
DECsystem. . . . . 53
value=15 . . . . . 53
```

Note, in the last example, that if the HYPHENATE flag (=) were enabled only the word "value" would have been entered in the index (refer to 5.2.2.1).

Example 4:

In the next example, CAPITALIZE flags are used to uppercase certain words for the index:

entering:

```
.FLAGS CAPITALIZE
     .
     .
.b2;If I ><hear I >forget>;
.br;if I ><see I >remem|ber;
.br;if I ><use I >understand>.
        ><chinese >proverb
```

produces:

```
CHINESE . . . . . 8

Forget . . . . . 8
HEAR . . . . . 8

Proverb. . . . . 8
Remem|ber; . . . . . 8
Understand . . . . . 8
USE. . . . . 8
```

Note, in the last example, that if the BREAK flag (|) were enabled only a portion of the word "remember" (i.e., remem) would be entered in the index.

INDEX Flag Method Summary

It can be seen that the use of the INDEX flag method requires careful consideration of flag conditions and usage. For these reasons, exclusively creating an entire index via the storage command method is generally preferable.

The following provides a complete list of the flags that affect this method of specifying items:

FLAG	EFFECT
CAPITALIZE flag (<)	word to index in upper case.
UPPERCASE flag (^)	next letter to index in upper case.
LOWERCASE flag (\)	next letter to index in lower case.
ACCEPT flag (_)	next character to index "as is".
UNDERLINE flag (&)	next character to index underlined.
OVERSTRIKE flag (%)	previous character to index overstruck.
SPACE flag (#)	word terminator
INDEX flag (>)	word terminator
HYPHENATE flag (=)	word terminator
BREAK flag (|)	word terminator

Note that a space, tab, or end-of-line code, will also terminate the entry of characters into the index.

Finally, be aware that if an item is enhanced especially for the index the enhancement will also appear in the body of the document.

5.3 OUTPUTTING THE INDEX

The following material describes the two methods of outputting an index (i.e., output command or TCX program).

Only one method may be used at a time. However, one method may be substituted for the other by replacing the output command (.PRINT INDEX or .DO INDEX), in the .RNO file, with a .REQUIRE command (for the .RNX file), or vice versa.

5.3.1 Creating a One-Column Index

The use of the storage commands and the appropriate flag loads items into the buffer, but does not cause the index to be printed. To cause a one-column print-out, one of two available output commands (.DO INDEX or .PRINT INDEX) must be used. The selected command is inserted at a point in the document where the user desires the index be generated.

The header format of the printed index will differ slightly depending on the output command used. However, both commands cause the contents of the Index Buffer to be formatted, output, and then cleared.

5.3.1.1 .PRINT INDEX Command

This command causes a .BREAK, forces two blank lines, and outputs the entire content of the index buffer, in alphabetical order, as a one-column index.

Entries are printed against the left margin with regular line spacing, and a blank line is left between entries having a different first letter. Those entries requiring a reference are directly followed by a dotted line and an appended page number; additional page numbers are separated by commas and extended to the next line if necessary.

The .PRINT INDEX command is generally inserted at the end of the .RNO file to cause the generation of the index at that point.

The command may be formatted as follows:

```
.PRINT INDEX

.PX
```

Terminators: {./EL/;/!}

Characteristics:

- This command is only supported by TOPS-10, TOPS-20, and VAX/VMS versions of the DSR.

- If this command is entered in the input file and the /INDEX switch is used for processing the index will not be generated.

5.3.1.2 .DO INDEX Command

This command causes a .BREAK, forces a new page, outputs header information, and then prints the entire content of the Index Buffer as a one-column index (i.e., does a .PRINT INDEX).

Entries are printed against the left margin with regular line spacing and in alphabetical order, a blank line is left between entries having a different first letter. Those entries requiring a reference are directly followed by a dotted line and an appended page number, any additional page numbers are separated by commas.

When the .DO INDEX command forces a new page and outputs the header, the initial header line output contains the first index page number (Page Index-n), set against the right margin and followed by two blank lines. The command then centers and prints the text-parameter (textl), if it is supplied, in upper case. If a text-parameter is not supplied, the word "INDEX" is centered and printed in upper case. The title information is then followed by four blank lines which, in turn, are followed by the index items.

The .DO INDEX command is generally inserted at the end of the .RNO file to invoke the generation of the index at that point.

The command may be formatted as follows:

```
.DO INDEX textl

.DX textl
```

Terminators: {EL/;}

Characteristics:

- This command is only supported by TOPS-10, TOPS-20, and VAX/VMS versions of the DSR.

- If this command is entered in the input file and the /INDEX switch is used for processing the index will not be generated.

5.3.2 Creating a Two-Column Index

During DSR processing, and in conjunction with the use of a DSR utility program (TCX), the index can be output in two-column format.

The operation is initiated during the processing of the .RNO file by using the /INDEX switch. In this manner, index buffer content is collected in a separate intermediate file (.BIX), for processing by TCX.

Two-Column Index (TCX) Program

The TCX program uses the .BIX file as input to create a separate .RNO type file that is defined by the program as type .RNX.

When the program is run, the following may be specified:

1. The number of entries desired per index page, to enhance the readability of the index.

2. References to running page numbers (refer to 3.1.1.5) instead of section oriented numbers.

The TCX utility program is run in a similar manner and uses the same form of input as the Table Of Contents (TOC) program (refer to Chapter 6).

Using the .RNX File

By creating an independent file (.RNX) from buffer content, and indirectly inserting it (via .REQUIRE "filnam.RNX") at the end of the .RNO file, a two-column index will be output. Moreover, the .RNX file may be independently processed to create an output file (.MEX) that contains the entire index.

Creating the Required Files

The procedure for creating the required files is as follows:

- To create the .BIX file: Run the DSR program to create a .MEM file and specify the index switch (/INDEX:filnam) on the command line. If no file name is specified, the name of the .RNO file is assumed.

- To create the .RNX file: Run the Two-column Index (TCX) program and input the .BIX file on request; on completion (FINISHED) the .RNX file is produced.

- To create the .MEX file: Run the DSR program using the .RNX file as input; the .MEX file will be created containing the two-column-index.

- To create a .MEM file containing a two-column-index: include the .RNX file in the .RNO file via a .REQUIRE command and run the DSR program, using the .RNO file as input.

For examples of the manner in which the DSR command lines are used to produce both the .BIX and .RNX input files, and the .MEX output file, refer to section 7.4.3.1.

5.3.3 Order of Index Entries

When a one- or two-column index is output, all items are printed in alphabetical order, for each separate level of entry, and all identically spelled items are grouped together. However, based on the possible enhancements (i.e., case, overstriking, underlining, bolding) imposed on an entry, identically spelled items will be listed in the following order:

1. Capitalized, overstruck, underlined, and bolded entries.
2. Capitalized, overstruck, and underlined entries.
3. Capitalized, overstruck, and bolded entries.
4. Capitalized and overstruck entries.
5. Capitalized, underlined, and bolded entries.
6. Capitalized and underlined entries.
7. Capitalized and bolded entries.
8. Capitalized entries.
9. Lower-cased, overstruck, underlined, and bolded entries.
10. Lower-cased, overstruck, and underlined entries.

11. Lower-cased, overstruck, and bolded entries.

12. Lower-cased and overstruck entries.

13. Lower-cased, underlined, and bolded entries.

14. Lower-cased underlined entries.

15. Lower-cased bolded entries.

16. Lower-cased entries.

Note that the order of entry for overstruck characters is based on the enhanced nature of the first (or overstruck) character and not on the second (or overstriking) character.

5.4 MISCELLANEOUS INDEXING COMMANDS

An adequate index may be created with the commands previously described; however, the following commands provide for such things as:

- Aiding in the creation of a subsectioned index via the .NUMBER INDEX command.

- Controlling the recognition of the SUBINDEX flag, or the entire range of indexing commands and flags, which may be independently or collectively disabled (.NO FLAGS SUBINDEX, .DISABLE INDEXING) and re-enabled (.FLAGS SUBINDEX, .ENABLE INDEXING).

Although these commands are seldom required, the following provides an example of usage.

Assume that an input file contains provisions for the creation of an index and the user desires to observe only a portion of the items specified in the file. For such an operation the .DISABLE and .ENABLE INDEXING commands could effectively be used to mask the file and restrict the output to certain items.

5.4.1 .NUMBER INDEX

This command re-enables page numbering and causes the word "Index" to appear as a prefix for each page number of the index (i.e., Index-n).

The .NUMBER INDEX command generally precedes a .PRINT INDEX command to provide prefixed page numbering for those indices created without header information. However, the initial index page number evoked will be a continuation of current page numbering (i.e., page numbering is not reset).

In addition, the command causes all subsequent header level numbers to be prefixed with the word "INDEX" (i.e., Index.1, Index.1.1, etc.).

The command may be formatted as follows:

```
.NUMBER INDEX

.NMNDX
```

Terminators: {.|EL|;|!}

5.4.2 .NO FLAGS SUBINDEX and .FLAGS SUBINDEX

These commands disable and re-enable the recognition of the SUBINDEX flag. However, the .FLAGS SUBINDEX command can also be used to redefine the SUBINDEX flag character (refer to section 4.2 Flag Control Commands).

The commands may be formatted as follows:

Disable	Resume
.NO FLAGS SUBINDEX	.FLAGS SUBINDEX
.NFL SUBINDEX	.FL SUBINDEX

Default:

The SUBINDEX flag character is a right angle bracket (>) by default.

5.4.3 .DISABLE INDEXING and .ENABLE INDEXING

These commands respectively disable and re-enable the operation of all index related commands and flags.

When the disabling command is used all subsequent index related commands and flags will be ignored.

When the enabling command is used, the collecting of items will be resumed.

The commands may be formatted as follows:

Disable	Resume
.DISABLE INDEXING	.ENABLE INDEXING
.DIX	.EIX

Terminators: {.|EL|;|!}

CHAPTER 6

CREATING A TABLE OF CONTENTS

6.1 INTRODUCTION

During DSR processing, and in conjunction with the use of a DSR utility program (TOC), a table of contents can be automatically generated for any sectioned document.

The operation is initiated during the processing of the .RNO file by using the /CONTENTS switch. In this manner, and without any required modification to the .RNO file, table of contents information will be collected as a separate intermediate file (.BTC.), for processing by TOC.

The Table of Contents (TOC) Program

The TOC program uses the .BTC file as input, to create a separate .RNO type file (.RNT) for insertion in the document as a required file (i.e., via a .REQUIRE command).

When the program is run, the following may be specified:

1. Relative indenting for header levels 3 through 6, to enhance readability.

2. References to running page numbers (refer to 3.1.1.5) instead of section oriented page numbers.

Thus, the TOC utility program is run in a similar manner and uses the same form of input as the Two-Column Index (TCX) program.

Contents of .RNT

The .RNT file will contain the title, and all of the chapter, appendix, index, and header level information contained in the document.

All major headings (e.g., title and chapters) will be especially spaced, and formatted in upper case letters, while all Header Level 2's will be indented. Header levels 3 through 6 may also be indented, via affirmative response to a TOC query, during creation of the .RNT file.

6.2 CONSTRUCTING A TABLE OF CONTENTS

To produce a .RNT file, the intermediate file (.BTC) must first be created, via the DSR program, by adding a switch (/CONTENTS:filnam) to the command line.

When processing is completed, all formal section and subsection information will have been gleaned from the .RNO file and assembled as the .BTC file.

If no name is specified for .BTC (i.e., /CONTENTS), the name of the input file is assumed.

Table of Contents Enhancements

If the user desires to influence the formatting of the table of contents, the information accumulated by the .BTC file may include DSR commands that are reserved for insertion in the file via .SEND TOC commands. These commands are inserted in the .RNO file to affect such things as vertical or horizontal spacing within the table.

For example, if the user desires to increase the vertical spacing before a Header Level One, commands may be inserted as follows:

```
.SEND TOC.BLANK2
.HL1 DSR COMMANDS
```

The ultimate effect is the insertion of a .BLANK2 command, at the designated point, in the .RNT file.

6.3 OUTPUTTING A TABLE OF CONTENTS

When .RNT is created, the file can be used not only to output the table of contents within the document but to output the table independently.

To output the table within the document, the .RNT file is indirectly inserted at the top of the .RNO file via command (.REQUIRE "filnam.RNT"). The table will then be output during the normal processing of the document via the DSR.

To output the table as an an independent document, the .RNT file itself is separately processed via the DSR and a .MEM type file (.MEC) is created. The table is then output via DSR processing.

For an example of the use of the /CONTENTS switch and the TOC program, and the manner in which the .BTC, .RNT and .MEC files are produced, refer to section 7.4.13.

6.4 TABLE OF CONTENTS COMMANDS

6.4.1 .DISABLE TOC and .ENABLE TOC

These commands disable and enable the generation of a table of contents by controlling the content of the .BTC file.

The creation of a table of contents is enabled by default. Therefore, if the command line switch (/CONTENTS) is used a .BTC file will be created. However, if .DISABLE TOC is used, the .BTC file cannot collect table of contents information until the feature is re-enabled.

The command may be formatted as follows:

Disable	Resume
.DISABLE TOC	.ENABLE TOC
.DTC	.ETC

Terminators: {.|EL|;|!}

Default:

.ENABLE TOC is the default condition.

6.4.2 .SEND TOC

This command causes the text (textl), directly following a number parameter (n), to be appear in the table of contents file (.RNT).

The text may be a string of DSR commands, flags, or any information that the user desires to insert in the table for enhancement purposes.

The text may be inserted in the .RNO file before the header levels or section titles the user desires to enhance.

The value of the number-parameter depends on the version of the TOC program used. For the TOC version associated with the DSR, a value is not required; this is the same as specifying a value of zero (0). However, if a zero is entered and textl contains a leading digit a comma-separator must be unserted (0,textl).

The command may be be formatted as follows:

```
.SEND TOC n,textl

.STC n,textl
```

Terminators: {.|EL|;|!}

CHAPTER 7

RUNNING THE DSR PROGRAM

7.1 INTRODUCTION

This chapter describes how to run the DSR program on each of the systems for which it is currently available (e.g., TOPS-10, TOPS-20, VAX/VMS, etc.).

The strategy used in the structuring of this material, initially provides a description of the procedures required to run the TOPS-10 version of DSR. Descriptions for running other versions are then provided in terms of modifications to the TOPS-10 procedures.

To distinguish between program responses, echoed on the terminal, and typed-in operator responses, the common convention of underlining computer output is used.

7.2 DSR CONTROL LANGUAGE

Running system-related (e.g., TOPS-10, TOPS-20, VMS, etc.) versions of DSR requires the use of appropriate monitor commands and DSR command lines.

The DSR command lines are instructions that are entered on the terminal to direct the operation of the DSR program (e.g., specify input/output files, affect global enhancements, etc.). As such, they should not be confused with the DSR formatting commands entered in the input file.

The control language consists of the following:

- Monitor commands (e.g., R RUNOFF), for accessing the DSR program.
- File specifications (e.g., dev:filnam.ext), for the command line, to select the file(s) to be processed and the I/O devices to be used.
- Optional switches (e.g., /DRAFT), for the command line, to evoke certain output file formatting features.

7.2.1 Rules for DSR Switches

Document formatting via the switches allows the user to do such things as: alter the position of the text on all pages of the document, control special effects (e.g., underlining, bolding), and create an index or table of contents automatically.

Note that the switches overide any conflicting DSR commands and/or flags imbedded in the input file.

The following rules apply to the use of command line switches:

- The user may append as many switches to a line as desired, as long as no operational conflict is incurred.
- Switch names may be entered in any case (i.e., upper, mixed, or lower).

- Switch names may be entered in a truncated form (e.g., /UNDERLINE: to /UND:).

- No spaces are allowed between the switch symbol (/) and a switch name, or between switch names and parameters (e.g., /FORMSIZE:58).

7.2.2 Using other Extensions or Types

Input file names that are not of the type (or extension) .RNO may also be used in the command line, if the complete name and extension are typed. When this is done the following defaults are evoked: (1) The output file name will be the same as the input file name. (2) The output extension (or type) will be a function of the input extension, as follows:

INPUT	OUTPUT
.RNB	.BLB
.RNC	.CCO
.RND	.DOC
.RNE	.ERR
.RNH	.HLP
.RNL	.PLM
.RNM	.MAN
.RNO	.MEM
.RNP	.OPR
.RNS	.STD
.RNT	.MEC
.RNX	.MEX
(none)	.MEM
(other)	.MEM

7.2.3 Control Language Syntax

Since control language syntax is system-dependent, the following material separately describes the commands required to process a file on each available system.

7.3 TOPS-10 CONTROL LANGUAGE

This section describes how to use DECsystem-10 to instruct DSR to: access an input file, produce a formatted output file, and direct the output either to storage or the terminal.

A complete command line format for TOPS-10 is as follows:

```
output-file-spec=input-file-spec/switch1/switch2.../switchn
```

7.3.1 Accessing DSR

When the TOPS-10 log-in process is complete, the monitor outputs a prompt character (.) to indicate it is ready for a command.

The user responds by typing R RUNOFF followed by a carriage return (CR):

```
.R RUNOFF<CR>
```

7.3.2 Output to Disk

The DSR program now prompts the user with an asterisk (*), and the user responds by typing the name of the input file, followed by a carriage return.

```
.R RUNOFF
*TEST<CR>
```

The DSR program assumes a .RNO extension. Therefore, if the input file already has an extension of .RNO only the file name need be specified, as in the example.

If an output file name is not specified, the DSR uses the input name for the output, and if an output device or input device is not specified, the disk is used for both purposes by default. In this manner, the type-in of the command line is abbreviated.

In any case, the input file (TEST.RNO) is processed and an output file is created, for which DSR provides an output file extension (TEST.MEM).

If no DSR errors are detected the program responds as follows:

```
.R RUNOFF
*TEST
No errors detected by version 1(71) of RUNOFF
25 pages written to DSK:TEST.MEM
*
```

The user may now return to monitor mode via a Control-C, and the output file may be printed:

```
*^C
.PRINT TEST.MEM
```

If a DSR error is detected a typical response is as follows:

```
.R RUNOFF
*TEST
%RNFJEC Junk at end of command: ".lm0rm70"
  on output page 1: on input line 8 of page 1 of file "DSK:TEST.RNO"
2 lines of messages
*****ERRORS DETECTED DURING PROCESSING*****
25 pages written to DSK:TEST.MEM
*
```

If DSR errors occur, refer to the list of error messages in Appendix B, correct the error with the editor and re-process the file.

7.3.3 Output to Terminal

If the user desires to display the processed output file on the terminal (TTY:), instead of writing it to a file, the user may type the following:

```
.R RUNOFF
*TTY:=TEST.RNO<CR>
(TEST.MEM is printed on terminal)
*^C
.
```

By specifying the terminal itself as the output device for the processed file, the effects of current modifications to the input file can be observed.

7.3.4 Input from Terminal

If prior to processing the .RNO file, the user desires to affect the output via additional commands, the user may specify the terminal (TTY:) as the input device, as follows:

```
.R RUNOFF
*TEST.MEM=TTY:<CR>
.SPACING2<CR>
.REQUIRE"NEST.RNO"<CR>
^Z
```

Note that the input from the terminal is terminated with a Control-Z (^Z), instead of a Control-C (^C), in order to provide an end-of-file. In addition, the "no errors detected" information will be output.

7.3.5 Input-Output to Terminal

If the user desires to experiment with various DSR commands and/or switches, and immediately analyze the results, the user may specify the terminal (TTY:) as both the input and the output device, as follows:

```
.R RUNOFF
*TTY:=TTY:<CR>
```

At this point, the user may enter DSR commands, along with the text to be formatted. The DSR commands and text are entered in the same manner as they would be if they were being entered in an input file.

```
.R RUNOFF
*TTY:=TTY:/UNDERLINE:SEPARATE
.DISPLAY CHAPTER RU<CR>
.NUMBER CHAPTER 5<CR>
.CHAPTER ^&dsr switches\&<CR>
```

(the display is as follows)

```
.
.
CHAPTER V

DSR SWITCHES
--- --------

^C

.
```

7.4 TOPS-10 COMMAND LINE SWITCHES

This section provides descriptions of the DSR command line switches available to DECsystem-10 users.

As with the DSR commands described in Chapter 3, the switches are arranged in general-usage groups.

7.4.1 /SIMULATE and /NOSIMULATE

These switches enable and disable the simulation of form feeds via line feed generation. The default is /NOSIMULATE.

Normally DSR skips to the top of a page via a form feed. If /SIMULATE is used the generation of form feeds is disabled. However, the form feed affect is accomplished by the generation of a sufficient number of blank lines to effect a skip to the top of each new page.

The /SIMULATE switch is generally used with devices that have no inherent form feed capability.

7.4.2 /FORMSIZE:number

This switch indicates, via its parameter, how many lines of text a page can accommodate.

7.4.2.1 /FORMSIZE and /NO SIMULATE

If /FORMSIZE is used and /SIMULATE is not specified, the default print capacity is 60 lines. Therefore, /FORMSIZE need only be used to indicate a capacity other than 60. Moreover, since external form feed control is also assumed, the capacity defined will be equal to the value specified by the parameter minus the number of lines reserved for spacing (top and bottom of page) by the device.

For example, assume the /FORMSIZE switch is used for a form with a 66 line capacity and the parameter specifies a value of 58. With these assumptions, a capacity of 50 lines of text per page is defined while 8 lines are reserved for top and bottom of the page spacing via the form feed mechanism (see Figure 3-1a).

7.4.2.2 /FORMSIZE and /SIMULATE

When the /SIMULATE switch is used the default print capacity is 66 lines. Therefore, /FORMSIZE need only be used to indicate a capacity other than 66.

For example, with /FORMSIZE:62 and /SIMULATE used, the parameter value specifies a capacity of 62 lines of text per page, while form feed control is simulated (see Figure 3-3).

Examples of the use of the switch follow:

```
*TEST.RNO/FORMSIZE:58

*TEST.RNO/FORMSIZE:62/SIMULATE
```

7.4.3 /PAUSE and /NOPAUSE

The /PAUSE switch causes a temporary halt to occur, and the terminal bell to ring, following the output of a page; the arbitrary depression of a character on the keyboard resumes the processing of output (without causing a typeout). The default condition is /NOPAUSE.

Using the /PAUSE switch allows single sheets of paper or multilith masters to be loaded, following each page of output, on I/O devices such as the DIABLO terminal.

Examples of the use of the switches follows:

```
*TTY:=TEST.RNO/PAUSE

*TTY:=TEST.RNO/NOPAUSE
```

7.4.4 /PAGES:"ranges"

This switch limits the output generated to the page(s) specified by the "ranges" parameter.

7.4.4.1 Range-Parameter Conventions

A range of page numbers is specified by start and end numbers that are separated by a colon and enclosed in quotes:

```
/PAGES:"start:end"
```

If only a start page number is specified, output will begin at the specified page and continue until the end of the file:

```
/PAGE:"start"
```

If more than one range of page numbers is specified the ranges are separated by commas and the entire string is enclosed in quotes:

/PAGES:"start1:end1,start2,start3:end3,...etc"

If only one page is desired, "start" and "end" must be the same number:

/PAGE:"5-30:5-30"

For example, the following specifies output from Section 4, page 12, through a single page of the appendix and six pages of the index:

/PAGES:"4-12,A-6:A-6,Index-5:Index-10"

7.4.4.2 Default Display Effect

The test for the requested page(s) is made without regard to .DISPLAY command effects, therefore, the ranges must define the page number(s) in terms of the default display characteristics.

For example: Appendix B, Page B-6 must be specified as /PAGE:"B-6", while Chapter V, Page V-13 (the result of a roman numeral display command) must be specified as /PAGE:"5-13".

7.4.4.3 Permissable Short-cuts

The partial specification of a range is possible if the user desires that the output occur at the start of an appendix or the index.

For an entire appendix only the letter is required (i.e., /PAGES:"A") and for an entire index only the word is required (i.e., /PAGES:"Index"); partial specification is not otherwise possible.

Examples of the use of the switches follows:

*TTY:=TEST.RNO/PAGE:"12"

*TTY:=TEST.RNO/PAGE:"2-12"

*TEST.RNO/PAGES:"2-9:2-9,A-1:A-5c"

7.4.5 /DOWN:number

This switch causes a specified number of blank lines to be generated, prior to the header information, at the top of each page. However, the number of blank lines specified by the /DOWN switch consequently affects the the maximum number of lines of text (and header information) that can be output on a page.

For example, if a /DOWN:10 switch is used and a .PAGE SIZE of 58 lines has been specified, the number of down lines desired will be subtracted from the number of text lines specified. Consequently, each page will contain a maximum of 48 lines of text and header information, while 10 blank lines will occur before the title line.

If the /DOWN switch is used and a parameter of zero (0) is specified no blank lines are invoked, except those imposed by the form feed control of a device.

If the /DOWN switch is used and no parameter is specified a default value of four (4) is invoked.

An example of the use of the switch follows:

```
*TEST.RNO/DOWN:10
```

7.4.6 /RIGHT:number

This switch causes the text on each page (including header information) to be shifted to the right by the number of spaces specified by the number parameter.

If the /RIGHT switch is used and no parameter is specified a default value of five (5) spaces is invoked.

If the /RIGHT switch is used and a parameter of zero (0) is specified no shift occurs.

An example of the use of the switch follows:

```
*TEST.RNO/RIGHT:8
```

7.4.7 /SEQUENCE and /NOSEQUENCE

These switches enable and disable the generation of input file sequence numbers in the output file; the default is /NOSEQUENCE.

For SOS type editors, which generate edit line-numbers and edit page-numbers in the input file (i.e., 00100/1 00200/1, etc.), the /SEQUENCE switch causes similar numbering to appear in the output.

The numbers appear in the left margin, at the beginning of each associated line of output.

If the editor (e.g., TECO) does not generate line-numbers in the input file, sequential numbers will still be generated, but without leading zeroes.

This feature can be extremely useful in editing documents. For example, the precise effect of blank line and/or line spacing insertions can be readily determined.

An example of the use of the switch follows:

```
*TTY:=TEST.RNO/SEQUENCE
```

7.4.8 /UNDERLINE:mode and /NOUNDERLINE

These switches enable and disable the underlining of text, and control how underlining will be accomplished (i.e., mode).

7.4.8.1 Underline Print Modes

There are four separate modes used to affect the underlining of a print-out:

- Line Mode
- Backspace Mode
- Separate Mode
- Replacement Mode

Line Mode

The Line Mode uses an underscore character (_) to underline text on an entire line, before line spacing can occur, by sending a carriage return to the device followed by such horizontal spacing as is required to reach the specified text. This mode is used by default.

Backspace Mode

The Backspace Mode assumes that the underline character, generally an underscore (_), is a spacing character for the output device and, therefore, backspaces and underlines each specified character as it is printed. This mode of operation is specified by the /BACKSPACE switch (refer to 5.3.2.3).

Separate Mode

The Separate Mode can be invoked for any device. In this mode underlining does not occur on the same line, but on the line below the text. In Separate Mode a dash (-) is used as the underline character instead of an underscore; the mode is invoked as follows:

```
/UNDERLINE:SEPARATE
```

Replacement Mode

The Replacement Mode uses two /UNDERLINE switch parameters, either separately or together, to specify a replacement for the underline character (by default an underscore):

```
/UNDERLINE:ch

/UNDERLINE:NONSPACING

/UNDERLINE:(NONSPACING,ch)
```

The specified character can be a spacing character (i.e., *, #, etc.) or a non-spacing character (e.g., octal code).

If a spacing character is used, it must be enclosed in quotes (:"ch"):

```
/UNDERLINE:"*"
```

If a non-spacing character or code is used, quotation marks are not allowed and the octal equivalent of the character must be specified.

```
/UNDERLINE:(NONSPACING,7)
```

An alternative method for coding the same string is as follows:

```
/UNDERLINE:NONSPACING/UNDERLINE:7
```

In the last example, the octal code (7) causes the terminal bell to ring (instead of underlining) every time a character to be underlined is encountered.

7.4.8.2 Disabling Underlining

The underlining of output can be disabled in the following ways:

```
/NOUNDERLINE

/UNDERLINE:NONE

/UNDERLINE:0
```

Examples of the use of the switch follow:

```
*TEST.RNO/UNDERLINE:SEPARATE

*TEST.RNO/UNDERLINE:"="

*TEST.RNO/UNDERLINE:0
```

7.4.9 /BOLD:number and /NOBOLD

These switches are used to enable and disable the bolding feature. In addition, a parameter can be used with the /BOLD switch to specify the number of times the text will be over-printed (i.e., bolded).

The default value for the parameter is one (1), and if a zero (0) is typed the effect is equivalent to using the /NOBOLD switch.

Examples of the use of the switches follow:

```
*TEST.RNO/BOLD

*TEST.RNO/BOLD:4

*TEST.RNO/BOLD:0

*TEST.RNO/NOBOLD
```

7.4.10 /BACKSPACE

This switch is used to direct DSR to use the BACKSPACE character to accommodate three special effects:

- The Backspace Mode of underlining (refer to 7.3.2.1).

- The bolding of characters (refer to 4.1.3.4).

- A multiple-overstrike method of creating special characters.

For, example if the input file contains a multiple-overstrike of the form:

=%/%\%^

unless the /BACKSPACE switch is used, only the final overstrike (i.e., =%^) can occur.

Using the switch also automatically enables the Backspace underline mode (i.e., /UNDERLINE:BACKSPACE) previously described.

An example of the use of the switch follows:

*TEST.RNO/BACKSPACE

7.4.11 /CHANGE:ch and /NOCHANGE

These switches enable and disable the appearance of change-bars (|) in the output file.

Using the /CHANGE switch to enable change-bars for an output file is the same as entering an .ENABLE BAR command at the beginning of its input file (refer to 3.3.7.5).

7.4.11.1 Replacing the Default Character

A character parameter may be used with the /CHANGE switch to specify a replacement for the change-bar character:

/CHANGE:ch

The specified replacement can be a spacing character (e.g., *, #, etc.) or a non-spacing character (e.g., octal code).

If a spacing character is used, it must be enclosed in quotes (:"CH"):

/CHANGE:"?"

If a non-spacing character code is used quotation marks are not allowed:

```
/CHANGE:7
```

In the last example, the octal code (7) causes the terminal bell to ring every time an altered line of output is encountered.

7.4.11.2 Disabling Change-Bars

The change-bar feature may disabled in the following ways:

```
/CHANGE:0

/NOCHANGE
```

Note that disabling change-bars overides any .ENABLE BAR command in the file.

Examples of the use of the switch follow:

```
*TEST.RNO/CHANGE:7

*TEST.RNO/CHANGE:"*"

*TEST.RNO/NOCHANGE
```

7.4.12 /INDEX:file-spec

This switch is used to create a special intermediate index file for the generation of a two-column index. Unless otherwise specified, the intermediate (binary) file will have an extension of .BIX.

The .BIX file is used as input to a DSR utility program (TCX) which, in turn, uses the file to generate a .RNO type input file (.RNX) that is acceptable to DSR for processing.

The .RNX file may be inserted into the main .RNO file via a .REQUIRE command or independently processed and printed as a .MEM type output file (.MEX) via RUNOFF.

When a .BIX file is created from a .RNO file (e.g., TEST.RNO) the name of the .RNO file is assumed for the .BIX file (e.g., TEST.BIX), unless a different file name is specified by the switch parameter (e.g., /INDEX:IND1).

An example of the use of the switch follows:

```
.R RUNOFF<CR>                                   !access DSR program.
*TEST.RNO/INDEX<CR>                             !generate .BIX file
No errors detected by version 1(71) of RUNOFF
25 pages written to DSK:TEST.MEM

.R TCX<CR>                                      !access TCX program.
TCX, version 1(16)
Specify input file:
TEST.BIX<CR>                                    !generate .RNX file.
Running page counters? (Y or N)
N<CR>                                           !no running numbers
Specifying number of index lines per page:
50<CR>                                          !restrict to 50.
FINISHED                                        !.RNX generated.

.R RUNOFF<CR>                                   !access DSR program.
*TEST.RNX<CR>                                   !produce .MEX
No errors detected by version 1(71) of RUNOFF
25 pages written to DSK:TEST.MEX

.
```

For details concering the creation of indices refer to Chapter V CREATING AN INDEX.

7.4.13 /CONTENTS:file-spec

This switch is used to create a special binary table of contents file for the generation of a table of contents; unless otherwise specified the file will have an extension of .BTC.

The .BTC file is used as input for a DSR utility program (TOC) which, in turn, uses the file to generate a .RNO type input file (.RNT) that is acceptable to the DSR for processing.

The .RNT file may be inserted into the main .RNO file via a .REQUIRE command or independently processed and printed as a .MEM type output file (.MEC) via RUNOFF.

When a .BTC file is created for a .RNO file (e.g., TEST.RNO) the name of the .RNO file is assumed for the .BTC file (e.g., TEST.BTC), unless a different file name is specified by the switch parameter (e.g., /CONTENTS:TOC1).

An example of the use of the switch follows:

```
.R RUNOFF<CR>                                   !access DSR program.
*TEST.RNO/CONTENTS<CR>                          !generate .BTC file.
No errors detected by version 1(71) of RUNOFF
25 pages written to DSK:TEST.MEM

.R TOC<CR>                                      !access TOC program.
TOC, version 1(14)
Specify input file:
TEST.BTC<CR>                                    !generate .RNT file.
Varying header-level indent?(Y or N)
Y<CR>                                           !indent .HL's 3 - 6.
Running page counter?  (Y or N)
N<CR>                                           !no running numbers.
FINISHED                                        !.RNT generated.

.R RUNOFF<CR>                                   !access DSR program
*TEST.RNT<CR>                                   !produce .MEC.
No errors detected by version 1(71) of RUNOFF
25 pages written to DSK:TEST.MEC

.
```

For further details concerning the creation of a table of contents refer to Chapter 6 CREATING A TABLE OF CONTENTS.

7.4.14 /DEBUG:echo

This switch traces the operation of any or all of the DSR commands, defined by the parameter (i.e., CONDITIONALS, FILES, INDEX, CONTENTS, ALL), by echoing each execution in the output file.

7.4.14.1 CONDITIONALS

This parameter causes all conditional commands (.IF, .IFNOT, .ELSE, .ENDIF) in the .RNO file to be ignored and the draft flags to appear in the .MEM file (refer to section 3.6.6 .VARIABLE Command).

7.4.14.2 FILES

This parameter causes all .REQUIRE commands (refer to section 3.6.3) in the .RNO file to be echoed in the .MEM file.

7.4.14.3 INDEX

This parameter causes all index items in the .RNO file to be echoed in the .MEM file, with each item appearing before the line of text with which it is associated.

All items that are a result of the .INDEX command and/or the INDEX flag are labelled with the word ".INDEX", while the results of .ENTRY commands are labelled with the word ".ENTRY".

7.4.14.4 CONTENTS

This parameter causes all .SEND TOC commands in the .RNO file to be echoed in the .MEM file.

7.4.14.5 ALL

This parameter specifies all four parameter possibilities, and all of the operations just described are performed simultaneously.

Examples of the use of the switch follow:

```
*TEST.RNO/DEBUG:CONDITIONALS

*TEST.RNO/DEBUG:FILES

*TEST.RNO/DEBUG:INDEX
```

```
*TEST.RNO/DEBUG:(CONTENTS,INDEX)

*TEST.RNO/DEBUG
```

7.4.15 /VARIANT:name

This switch controls the execution of the conditional commands (.IF, .IFNOT, .ELSE, .ENDIF) by specifying, via the parameter, the name(s) of the segment(s) to be processed (refer to section 3.6.5 .IF, .IFNOT, .ELSE, and .ENDIF).

Examples of the use of the switch follow:

```
*TEST.RNO/VARIANT:A

*TEST.RNO/VARIANT:BETA

*TEST.RNO/VARIANT:"A,B,C"
```

Note in the last example that if more than one parameter name is used the string must be enclosed in quotes.

7.4.16 /MESSAGES:destination

This switch specifies the destination of all DSR error messages via a parameter (OUTPUT or USER).

Messages are both sent to the .MEM file and displayed on the terminal, by default.

If OUTPUT is specified, messages are sent to the .MEM file only.

If USER is specified, messages are displayed on the terminal only.

The default condition is /MESSAGES:(OUTPUT,USER).

Examples of the use of the switch follow:

```
*TEST.RNO/MESSAGES:OUTPUT

*TEST.RNO/MESSAGES:USER
```

7.4.17 /HELP

This switch causes the contents of the DSR help file (.HLP) to be displayed on the terminal.

The .HLP file contains DSR command line information for the user.

An example of the use of the switch follows:

```
.R RUNOFF<CR>
*/HELP<CR>
(.HLP file is displayed)
```

7.5 TOPS-20 CONTROL LANGUAGE

to be supplied

7.6 TOPS-20 COMMAND LINE SWITCHES

to be supplied

7.7 VMS CONTROL LANGUAGE

This section describes how to use VAX/VMS to instruct DSR to: access an input file, produce a formatted output file, and direct the output to either the disk (for storage and printing) or the terminal (for display).

A complete command line format for VMS is as follows:

```
input-file-spec/switch1/switch2.../switchn=output-file-spec
```

7.7.1 Accessing the DSR Program

When the VMS log-in process is complete, the monitor outputs a prompt character ($) to indicate command readiness.

The user responds by typing a space followed by RUNOFF, the name of the input file, and a carriage return (CR).

```
$ RUNOFF TEST.RNO<CR>
```

7.7.2 Output to Disk

The DSR program now processes the .RNO file and outputs a .MEM file to the disk under the same conditions previously described (refer to section 5.2).

If no DSR errors are detected the program responds and automatically returns to the monitor mode, at which point the .MEM file may be printed, as follows:

```
$ RUNOFF TEST.RNO
No errors detected by version 1.107 of RUNOFF
25 pages written to DBB1:[ELLIS]TEST.MEM;1
$ PRINT TEST.MEM
```

If a DSR error is detected the program will respond as follows:

```
$ RUNOFF TEST.RNO
%RNFJEC Junk at end of command: ".lm0rm70"
on output page 1; on input line 8 of page 1 of file "DBB1:TEST.RNO"
```

```
2 lines of messages
*****ERRORS DETECTED DURING PROCESSING*****
25 pages written to DBB1:[ELLIS]TEST.MEM
$
```

When DSR errors occur, the user may refer to the list of error messages in Appendix B.

Using Other Types

Input file name types that may be used are the same as previously described (refer to section 7.2.2 Using Other Extensions).

7.7.3 Output to Terminal

If the user desires to display the processed output file on the terminal (TT:), instead of saving it in a file, type the following:

```
$ RUNOFF TEST.RNO/OUTPUT:TT:<CR>
(TEST.MEM is displayed on terminal)
$
```

7.7.4 Input from Terminal

If, prior to processing the .RNO file, the user desires to output additional commands, the user may specify the terminal (TT:) as input, as follows:

```
$ RUNOFF TT:/OUTPUT:TEST.MEM<CR>
.REQUIRE"NEST.RNO"<CR>
^Z
```

Note that the input from the terminal is terminated with a Control-Z (^Z) to provide an end-of-file. In addition, the "no errors detected" information will be output.

7.7.5 Input-Output to Terminal

If the user desires to test individual DSR commands or switches, using immediate terminal display, the user may specify the terminal as both the input device and the output device, as follows:

```
$ RUNOFF TT:/OUTPUT:TT:<CR>
.FLAGS CAPITALIZE<CR>                    !enable capitalization
<capitalize the first word<CR>           !test text
.<CR>                                    !end-of-line and execute.
(the display is as follows)
          .
          .
CAPITALIZE the first word
```

7.8 VMS COMMAND LINE SWITCHES

The DSR command line switches are available to VMS as described for TOPS-10, with the following exceptions.

7.8.1 /FORMSIZE and /NOSIMULATE

The /FORMSIZE switch is only required if the paper cannot accommodate 66 lines of text.

With VMS there is no distinction made between /SIMULATE and /NOSIMULATE. VAX/VMS will not add blank lines to the top and bottom of the page.

7.8.2 /HELP

The /HELP switch is not supported; use the system HELP command.

7.9 RSX CONTROL LANGUAGE

To be supplied.

7.10 RSX COMMAND LINE SWITCHES

To be supplied.

APPENDIX A

DSR/RUNOFF COMPATIBLE FEATURES

A.1 INTRODUCTION

This material describes the use of certain features, commonly available with earlier versions of RUNOFF, that are supported by DSR for compatibility purposes only.

A.1.1 .PAPER SIZE Command

The .PAPER SIZE command is the same as the .PAGE SIZE command and may be used interchangeably.

The command may be formatted as follows:

```
.PAPER SIZE n1,n2

.PS n1,n2
```

A.1.2 .SUBINDEX Command

The .SUBINDEX command may be used interchangeably with the .INDEX command to more clearly define subdivided items in the source file. That is, the user may specify major items with the .INDEX command and (via flags) specify minor items with the .SUBINDEX command.

The command may be formatted as follows:

```
.SUBINDEX text2

.IX text2
```

A.1.3 The DRAFT Switch

The /DRAFT switch is supported by TOPS-10 and TOPS-20 only; moreover, using the /DRAFT switch is the same as using /DEBUG:ALL or /DEBUG.

A.1.4 The QUOTE Flag (_)

The QUOTE flag is the same as the ACCEPT flag and may be used interchangeably.

A.1.5 ENDFOOTNOTE and COMMENT Flag Usage

Since the exclamation mark (!) can be used, alone and in various combinations, as an ENDFOOTNOTE flag, command terminator, or COMMENT flag, a complete description of the use of the character follows.

A.1.5.1 The ENDFOOTNOTE Flag (!)

In certain versions of RUNOFF it is possible to terminate a footnote by the insertion of an ENDFOOTNOTE flag (!). With DSR this practice is not recommended; however, the flag is recognized by default and may be used with the following considerations.

The ENDFOOTNOTE flag is operative only if:

1. The flag is not collectively disabled (.NO FLAGS ALL).

2. The flag is not individually disabled (.NO FLAGS ENDFOOTNOTE).

3. A .FOOTNOTE command has been previously entered.

4. The flag is inserted in physical or logical column one.

Following the insertion of a .FOOTNOTE command, the entry of footnote information may be terminated by entering either an .END FOOTNOTE command or an ENDFOOTNOTE flag. If the flag is used, it must be inserted in column one (i.e., physical or logical) as a direct replacement for the command (refer to 2.1.3.4).

The flag may also be inserted within a multi-command string, as long as it is separated from the previous command by a semi-colon terminator (refer to 2.1.3.3), for example:

```
.SKIP.LM0;!
```

Note that the semi-colon terminator causes DSR to consider the next character as being in column one, regardless of where the character appears on the line.

When paired with the ACCEPT flag (_!), the flag character is taken as normal text.

A.1.5.2 .NO FLAGS and .FLAGS ENDFOOTNOTE

These commands disable and re-enable the recognition of the ENDFOOTNOTE flag.

The commands may be formatted as follows:

Disable	Resume
.NO FLAGS ENDFOOTNOTE	.FLAGS ENDFOOTNOTE
.NFL ENDFOOTNOTE	.FL ENDFOOTNOTE

Default:

The ENDFOOTNOTE flag character is an exclamation mark (!) by default.

A.1.5.3 The COMMENT Flag (!)

The COMMENT Flag can be used in place of the COMMENT keyword to allow the insertion of user comments in and for the .RNO file (i.e., the commentary will not appear in the output file).

When the flag is inserted in a multi-command string it becomes both the terminator of the previous command and the initiator of the commentary, for example:

```
.LMO.RM60!place comment here
```

The flag may be paired as follows:

- Following a CONTROL flag (.!), to introduce a comment at the start of a line:

```
.!place comment here
```

- Following an ACCEPT flag (_!), which allows the character to be taken as normal text.

A.1.5.4 Summary and Examples

When the exclamation mark (!) character is used as a terminator for a DSR command it also acts as a COMMENT flag.

The dual usage of the character as both a terminator/COMMENT flag and an ENDFOOTNOTE flag should not be confused; the difference is:

As an ENDFOOTNOTE flag the character must be placed in logical or physical column one (i.e., following a semi-colon or an EL), while as a terminator/COMMENT flag the character must not be placed in column one.

The following examples depict the use of the exclamation mark as both a terminator/COMMENT flag and a replacement for a .COMMENT command.

```
.SKIP!commentary          !"c" is in logical column one
                          !the flag is not.

.!commentary              !"c" is in logical column one.
```

The next examples depict the use of the character as an ENDFOOTNOTE flag (i.e., a replacement for an .ENDFOOTNOTE command).

```
!{EL}                     !flag is in physical column one.

.SKIP;!                   !flag is in logical column one.
```

The following three examples are included to aid in the recognition of the character as a terminator/COMMENT flag and an ENDFOOTNOTE flag.

In this example, an ENDFOOTNOTE flag (!) replaces an .END FOOTNOTE command and is followed by a COMMENT flag (!):

```
.INDEX text;!!commentary{EL}
```

The next example is the same as the last except the flags are inserted at the beginning of a new line:

```
.INDEX text{EL}

!!commentary{EL}
```

This final example is the same as the last two except the COMMENT flag begins a new line:

```
.INDEX text;!{EL}
.!commentary
```

APPENDIX B

DSR MESSAGES

B.1 INTRODUCTION

There are four types of messages:

- o message text that is standard (error/non-error).
- o messages resulting from operator or user error.
- o messages resulting from DSR program failures.
- o message text that is standard for all errors.

The following conventions are used to describe the general nature of a DSR error message and, via the percent symbols, additional error-specific information.

```
?RNFXXX text"%x"
%RNFXXX text"%x"
```

where:

?	=	a fatal error message
%	=	a warning or advisory message
RNF	=	RUNOFF mnemonic
XXX	=	3-letter message mnemonic
text	=	message description

"%x" = can indicate the following:

%C = line number of an input file

%E = the name of a flag

%F = input-file-spec

%I = input file page number

%N = a correct value

%O = output-file-spec

%P = output file page number

%Q = name of footnote work file

%S = actual error string

%T = spec of a .BTC file

%V = DSR version number

%X = spec of a .BIX file

The following material initially describes the three types of error messages (i.e., user, program, standard) followed by the standard DSR message lines; where possible all messages are listed alphabetically.

B.2 DSR OPERATOR ERROR MESSAGES

The following messages specify operator or user induced errors.

1. %RNFBMS Bad margin specification: "%S"

   An illegal value has been used for a margin setting.

2. %RNFBVN Missing or illegal variable name: "%S"

   The name-parameter for a .VARIABLE, .IF, .IFNOT, .ENDIF, or .ELSE command is missing or illegal.

3. %RNFCEM Comma expected, missing: "%S"

   A required comma-separator is missing from the command string.

4. %RNFCJL Can't justify line

   The text and spacing is greater than the number of columns available between the margins.

5. %RNFCNF Character string expected, not found: "%S"

   A required character string is missing.

6. %RNFCNS Command not supported: "%N"

   The command is not supported by this version of the DSR; in this case %N is an internal number of interest only to DSR developers.

7. %RNFCOF Can't open footnote work file "%Q"

   The DSR cannot create the temporary file required to process the footnote information.

8. ?RNFCOI Can't open input file %F

   Input-file-spec erroneous or file is non-existent.

9. ?RNFCOO Can't open output file "%O"

   Output-file-spec erroneous, or write access is not allowed.

10. %RNFCOR Can't open required file %F

    Required file-spec erroneous or file is non-existent or read access is not allowed.

11. ?RNFCOT Can't open table of contents file %T

    The DSR cannot create the .BTC type file, required to generate a table of contents from the titles and header levels.

12. ?RNFCOX Can't open indexing file %X

    The DSR cannot create the .BIX type file, required to generate a two-column-index from the content of the index buffer.

13. %RNFCRF Can't read back saved footnotes.

    The DSR has successfully written footnote information to the work file but cannot read it back.

14. ?RNFCRP Can't recognize page on /PAGES switch

    A page range-parameter is illegally formatted.

15. %RNFCWF Can't write footnote file. (%N)

    DSR can open the footnote work file but cannot write into it; %N is a system error code.

16. %RNFDNS Detected an illegal .NO SPACE command

    A .NO SPACE command has been inserted following a command which issues a .BREAK.

17. %RNFDVN Duplicate variable name: "%S"

    An attempt has been made to use a name-parameter with a .VARIABLE command that has already been used with a previous .VARIABLE, .IF, .IFNOT, or .ELSE command.

18. %RNFDWF Date won't fit on line after subtitle

    Entered .DATE command; but the subtitle is too long and extends into the date field.

19. %RNFEFD .END FOOTNOTE doesn't terminate .FOOTNOTE

    An .ENDFOOTNOTE command is improperly positioned in the file; it does not follow associated footnote data.

20. %RNFELD .END LITERAL doesn't follow .LITERAL

    The .END LITERAL command is improperly positioned in the file; it does not follow the .LITERAL command.

21. %RNFEVL Too many variables: "%S"

    The maximum of 20 variables has been exceeded with the inclusion of the specified .VARIABLE or conditional command (.IF, .IFNOT, .ENDIF, or .ELSE); the command in error has been ignored.

22. %RNFFEL %E flag at end of line ignored

    This flag cannot occur at the end of a line.

23. %RNFFNA Flag not allowed in this context: "%S"

    A flag has been used which requires that text directly precede and/or follow it.

    A flag character has been replaced by another (via a .FLAGS type command) and the previous function of the replacement has not been disabled (via a .NO FLAGS type command).

24. %RNFFWF Footnotes won't fit on page.

    Too many footnotes, or too many lines in a footnote.

25. %RNFGFC Given footnote count incorrect. Correct value is %N

    Actual number of lines generated for the footnote is greater or less than the count specified.

26. %RNFIFT Illegal in FOOTNOTE: "%S"

    Attempted to use a command between .FOOTNOTE and .ENDFOOTNOTE that is illegal.

27. %RNFIIF %S ignored

    A control character code has been illegally entered (no .CONTROL CHARACTERS command) in the input file and will be ignored; the code value + 100 octal is displayed (%S).

In order to legalize the entry of such codes, the .CONTROL CHARACTERS command must first be entered.

28. %RNFILC Illegal command: "%S"

This command is illegal (keyword or parameters unrecognizable).

29. %RNFINI Improperly nested: "%S"

Conditional commands improperly matched (i.e., .IF/.IFNOT and .ELSE/.ENDIF).

30. %RNFINM Illegal number value: "%S"

Value is too large, too small, or illegal.

31. %RNFITC Index entry too complicated or long

The entry exceeds the limit of the index buffer for a single item.

32. %RNFITD .IF commands nested too deep: "%S"

Number of nested conditional commands (i.e., .IF, .IFNOT) exceeds the maximum limit of 10.

33. ?RNFIVS Illegal /VARIANT switch

Name-parameter for conditional commands missing or illegal.

34. %RNFJEC Junk at end of command: "%S"

Extraneous text (not a valid terminator) follows this command.

35. %RNFLDE Literal doesn't end with .END LITERAL: "%S"

.END LITERAL command does not immediately follow the final line of a counted .LITERAL command.

36. %RNFLTC Line too complicated: "%S"

Excessive use of bolding, underlining, overstriking, justifying and hyphenation on the line.

37. %RNFMEI Missing at least one .ENDIF command

End-of-file has been detected and at least one conditional command (.IF/.IFNOT) is not legally terminated.

38. %RNFMFN Number illegal or malformed: "%S"

Too many digits used for number-parameter or I/O.

39. %RNFMFS Missing or improperly delimited file spec: "%S"

The file-spec-parameter for the .REQUIRE command is not enclosed in single or double quotes.

40. %RNFMNL Missing number or letter: "%S"

A required number or letter parameter is missing.

41. %RNFMQS Missing or improperly delimited quoted string: "%S"

Leading or trailing single or double quotes are missing from quoted-string-parameter.

42. %Another %N crossed margin or bad right indent attempts detected and accumulated. Now being reported

The above indicates that a specific number (%N) of similar horizontal spacing errors have been detected and are now being reported.

43. %RNFMRG Margins crossed, or attempted indent too far right

The right margin has crossed the left margin or the left margin has crossed the right margin, or an illegal indent has been specified.

44. ?RNFNFS No file specified

A file-spec was not specified on the DSR command line.

45. %RNFNIA Negative indent attempted

Attempted negative indent beyond left margin limit (.i.e, .LM0).

46. %Another %N negative indents detected and accumulated. Now being reported

    The above indicates a specific number (%N) of negative indents (beyond .LM0) have been detected and are now being reported.

47. %RNFNNA Negative number not allowed: "%S"

    A negative number has been illegally specified, or an illegal resultant negative value (due to addition or subtraction of values) has been detected.

48. %RNFNSF .END LIST/NOTE not in same file as .LIST/.NOTE: "%S"

    Indicates illegal attempt to start a list or note in one file and end in another.

49. %RNFNTD Files nested too deep: "%S"

    Indicates an attempt to nest required files more than 10 deep.

50. %RNFPWF Page number won't fit on title

    Indicates title (on first header information line) extends into page number field; therefore the page number won't fit.

51. %RNFQST Quoted string too long: "%S"

    A quoted-string-parameter contains too many characters.

52. %RNFRTL Input record too long: truncated "%S"

    Too many characters were entered on a single input line; excess has been discarded.

53. %RNFSKC .ENDIF/.ELSE not in same file as .IF/.IFNOT: "%S"

    Indicates illegal attempt to use associated conditional commands in separate files.

54. %RNFSTD Too many nested .NOTEs and/or .LISTs: "%S"

    Indicates an attempt to nest more than 20 notes or lists.

55. %RNFTAR No text allowed after .REQUIRE command: "%S"

    Indicates an attempt to insert commands or text directly following a .REQUIRE command parameter.

56. %RNFTFE Too few end commands

    A .END type command (e.g., .END LIST) is missing.

57. ?RNFTMP Too many page ranges on /PAGES switch

    More than 5 separate ranges have been specified with the switch.

58. %RNFTMT Too many tab settings; excess ignored: "%S"

    More than 40 separate settings have been used with a single .TAB STOPS command.

59. ?RNFTMV Too many /VARIANTs

    More than 20 variable names have been used with the switch.

60. %RNFTTL Text too long: "%S"

    Text-parameter for command (e.g., .TITLE , .CHAPTER, .NOTE, etc.) is too long, or contains excessive use of underlining, bolding and/or overstriking.

61. %RNFUDS Undefined symbol: "%S"

    The DSR has detected an unrecognizable symbol (e.g., $$name), the name of which is not defined.

62. %RNFUME Unmatched end command: "%S"

    A .END type command (e.g., .END NOTE) has been detected, but the associated start command (e.g., .NOTE) has not.

B.3 DSR INTERNAL ERROR MESSAGES

The following messages indicate program errors, and limitations, rather than operator or user errors.

1. %RNFFAB File aborted.

   The DSR has aborted the file for internal reasons.

2. %RNFIBO Input buffer overflow: "%S"

   Too long a character string has been entered in an internal line buffer.

3. %RNFILE ********INTERNAL LOGIC ERROR********(%S)

   The DSR has detected an internal failure at the location specified by the indicated string (%S).

4. %RNFSSR ****SCANT SAW RINTES!!!!

   DSR has detected a reserved File Separator (FS) code (octal 34), that is not allowed in DSR input files.

5. %RNFURE Unrecoverable error processing record %C on page %I of input file %F

   An unrecoverable I/O error has occurred, terminating the processing at the specified edit line (%C) and page number (%I) of the input file (%F).

6. %RNFURR Unrecognized request: "%N"

   DSR detected an unrecognizable request: internal failure.

7. %RNFXFL ******INDEX OVERFLOW, RESULTS UNDEFINED******

   During the construction of a one-column-index, the number of entries exceeded available memory; excess entries were discarded.

B.4 DSR STANDARD MESSAGE TEXT

The following is standard message text, some of which accompanies only error messages.

1. See command on input line %C of page %I of file %F

   This is a standard error message line which specifies the location of an error-related command in the input file.

   The message indicates: the name of the input file (%S) in which the command is located, and the file page number (%I) and line number (%C) on which it occurs.

2. on output page %P; on input line %C of page %I of file "%F"

   This is a standard error message line which specifies the location of an error string:

   The message indicates: the number of the page (%P) in the output file affected by the error and the name of the input file (%F) in which the error occurred, including the input page (%I) and line number (%C).

3. %N lines of messages

   This is a standard error message line which specifies the actual number of error message lines output.

4. *****ERRORS DETECTED DURING PROCESSING*****

   This is a standard error message line which indicates to the user that one or more errors have been detected.

5. No errors detected by version %V of RUNOFF

   This message indicates the successful processing of a file by the DSR, and the version (%V) of DSR used.

6. %N page written to %O

   This message indicates that one page, or none, (%N) has been written to the specified output file (%O).

7. %N pages written to %O

   This message indicates that a multiple number of pages (%N) have been written to the specified output file (%O).

APPENDIX C

DSR SYNTAX AND COMMAND LISTING

The DSR commands consist of four parts:

1. A CONTROL flag (.) which directly precedes the first keyword of a command; the flag is not shown in the listing.
2. A required keyword or keywords which specify a commands function.
3. Required or optional parameter(s) to provide additional functional information.
4. One of four possible Terminator characters to define the end of a command.

The following describes the symbols and conventions used to define command structure.

- The keyword(s) of a command is shown enclosed in braces ({}) to symbolize the choice of a legal abbreviation, a truncation, or the complete form of the keyword(s); each choice is separated by a vertical bar (|) which symbolizes the word "or".

  {Key|keywd|keyword}

- The legal abbreviation for a command is denoted by the first vertical bar (|).

  {Key|keyword}

- A legal truncation for a command is denoted by the second vertical bar (|).

  {Key|keywd|keyword}

- A required keyword space-separator is indicated by an at sign (@).

  {Keyword@Keyword}<param>

- Each parameter (param1, param2, etc.) is shown enclosed in angle brackets (<>).

  {Keyword(s)}<param1>,<param2>...

- The choice of a required terminator is also shown enclosed in braces ({.|;|EL|!}), with each possible selection separated by the vertical bar (|); however, if all four terminators can be used, the word "all" will appear in the braces.

  {Keyword(s)}<param>{all}

Parameter Symbols

All parameter types available to a command are shown and no distinction is made between required and optional parameters; for such usage refer to the command descriptions in Chapter 3.

The parameter symbols are as follows:

- number-parameter (n)
- count-parameter (c)
- text-parameter (text1 or text2)
- character-parameter (k)
- quoted-string-parameter (q)
- display-descriptor-parameter (y)
- name-parameter (name)

o draft-flag-parameter (flags)

The following is an alphabetical listing of all of the DSR commands:

{AX|APPENDIX}<textl>{all}

{AP|AUTOPARAGRAPH}{all} (no parameters)

{AST|AUTOSUBTITLE}{all} (no parameters)

{AT|AUTOTABLE}{all} (no parameters)

{BB|BEGIN@BAR}{all} (no parameters)

{B|BLANK}<n>{all}

{BR|@|BREAK}{all} (no parameters)

{C|CENTER|CENTRE}<n>{;|EL}<textl>{EL}

{CH|CHAPTER}<textl>{all}

{!|COMMENT}<textl>{;|EL}

{CC|CONTROL@CHARACTERS}{all} (no parameters)

{D|DATE}{all} (no parameters)

{DBB|DISABLE@BAR}{all} (no parameters)

{DBO|DISABLE@BOLDING}{all} (no parameters)

{DHY|DISABLE@HYPHENATION}{all} (no parameters)

{DIX|DISABLE@INDEXING}{all} (no parameters)

{DOV|DISABLE@OVERSTRIKING}{all} (no parameters)

{DTC|DISABLE@TOC}{all} (no parameters)

{DUL|DISABLE@UNDERLINING}{all} (no parameters)

{DAX|DISPLAY@APPENDIX}<y>{all}

{DCH|DISPLAY@CHAPTER}<y>{all}

{DLE|DISPLAY@ELEMENTS}<q><y><q>{all}

{DHL|DISPLAY@LEVELS}<y1><y2><y3><y4><y5><y6>{all}

{DNM|DISPLAY@NUMBER}<y>{all}

{DSP|DISPLAY@SUBPAGE}<y>{all}

{DX|DO@INDEX}<text1>{;|EL}

{ELSE<name>{all}

{EBB|ENABLE@BAR}{all} (no parameters)

{EBO|ENABLE@BOLDING}{all} (no parameters)

{EHY|ENABLE@HYPHENATION}{all} (no parameters)

{EIX|ENABLE@INDEXING}{all} (no parameters)

{EOV|ENABLE@OVERSTRIKING}{all} (no parameters)

{ETC|ENABLE@TOC}{all} (no parameters)

{EUN|ENABLE@UNDERLINING}{all} (no parameters)

E **END keyword abbreviation**

{EI|ENDIF<name>{all}

{EB|END@BAR}{all} (no parameters)

{END@FOOTNOTE}{all} (no parameters)

{ELS|END@LIST}{all} (no parameters)

{EL|END@LITERAL}{all} (no parameters)

{EN|END@NOTE}{all} (no parameters)

{ES|END@SUBPAGE}{all} (no parameters)

{Y|ENTRY<text1>{;|EL}

{FG|FIGURE}<n>{;|EL}

{FGD|FIGURE@DEFERRED}<n>{;|EL}

{F|FILL}{all} (no parameters)

{FT|FIRST@TITLE}{all} (no parameters)

FL **FLAGS keyword abbreviation**

{FLAGS<@ALL>}{all} (no parameters)

{FLAGS@BOLD}<k>{all}

{FLAGS@BREAK}<k>{all}

{FLAGS@CAPITALIZE}<k>{all}

{FLAGS@COMMENT}<k>{all}

{FLAGS@CONTROL}<k>{all}

{FLAGS@ENDFOOTNOTE}<k>{all}

{FLAGS@HYPHENATE}<k>{all}

{FLAGS@INDEX}<k>{all}

{FLAGS@LOWERCASE}<k>{all}

{FLAGS@OVERSTRIKE}<k>{all}

{FLAGS@PERIOD}<k>{all}

{FLAGS@QUOTE}<k>{all}

{FLAGS@SPACE}<k>{all}

{FLAGS@SUBINDEX}<k>{all}

{FLAGS@SUBSTITUTE}<k>{all}

{FLAGS@UNDERLINE}<k>{all}

{FLAGS@UPPERCASE}<k>{all}

{FN|FOOTNOTE}<n>{all}

HD **HEADER keyword abbreviation**

{HL|HEADERS@LEVEL}<n><text2>{all}

{HEADERS@LOWER}{all} (no parameters)

{HEADERS@MIXED}{all} (no parameters)

{HEADERS@<@ON>}{all} (no parameters)

{HEADERS@UPPER}{all} (no parameters)

{HY|HYPHENATION}{all} (no parameters)

{IF<name>{all}

{IN|IFNOT<name>{all}

{I|INDENT}<n>{all}

{X|INDEX<text2>{;|EL}

{J|JUSTIFY}{all} (no parameters)

{LO|LAYOUT}<n1><n2>{all}

{L|LEFT}<n>{all}

{LM|LEFT@MARGIN}<n>{all}

{LS|LIST}<n><q>{all}

{LE|LIST@ELEMENT}{all} (no parameters)

{LT|LITERAL}<n>{all}

{LC|LOWER@CASE}{all} (no parameters)

N **NO keyword abbreviation**

{NAP|NO AUTOPARAGRAPH}{all} (no parameters)

{NAST|NO AUTOSUBTITLE}{all} (no parameters)

{NAT|NO AUTOTABLE}{all} (no parameters)

{NCC|NO CONTROL@CHARACTERS}{all} (no parameters)

{ND|NO DATE}{all} (no parameters)

{NF|NO FILL}{all}	(no parameters)
{NFL|NO FLAGS<@ALL>}{all}	(no parameters)
{NO FLAGS@BOLD}{all}	(no parameters)
{NO FLAGS@BREAK}{all}	(no parameters)
{NO FLAGS@CAPITALIZE}{all}	(no parameters)
{NO FLAGS@COMMENT}{all}	(no parameters)
{NO FLAGS@CONTROL}{all}	(no parameters)
{NO FLAGS@ENDFOOTNOTE}{all}	(no parameters)
{NO FLAGS@HYPHENATE}{all}	(no parameters)
{NO FLAGS@INDEX}{all}	(no parameters)
{NO FLAGS@LOWERCASE}{all}	(no parameters)
{NO FLAGS@OVERSTRIKE}{all}	(no parameters)
{NO FLAGS@PERIOD}{all}	(no parameters)
{NO FLAGS@QUOTE}{all}	(no parameters)
{NO FLAGS@SPACE}{all}	(no parameters)
{NO FLAGS@SUBINDEX}{all}	(no parameters)
{NO FLAGS@SUBSTITUTE}{all}	(no parameters)
{NO FLAGS@UNDERLINE}{all}	(no parameters)
{NO FLAGS@UPPERCASE}{all}	(no parameters)

{NHD|NO HEADERS}{all} (no parameters)

{NHY|NO HYPHENATION}{all} (no parameters)

{NJ|NO JUSTIFY}{all} (no parameters)

{NNM|NO NUMBER}{all} (no parameters)

{NPA|NO PAGING}{all} (no parameters)

{NPR|NO PERIOD}{all} (no parameters)

{NSP|NO SPACE}{all} (no parameters)

{NST|NO SUBTITLE}{all} (no parameters)

{NT|NOTE}<textl>{EL}

NM **n keyword abbreviation**

{NMAX|NUMBER@APPENDIX}<c>{all}

{NMCH|NUMBER@CHAPTER}<c>{all}

{NMNDX|NUMBER@INDEX}{all} (no parameters)

{NMLV|NUMBER@LEVEL}<n1><n2><n3><n4><n5><n6>{all}

{NMLS|NUMBER@LIST}<n1<n2>{all}

{NMPG|NUMBER<@PAGE>}<c>{all}

{NMR|NUMBER@RUNNING}<c>{all}

{NMSPG|NUMBER@SUBPAGE}<c>{all}

{PG|PAGE}{all} (no parameters)

{PS|PAGE@SIZE"|PAPER@SIZE}<n1<n2>{all}

{PA|PAGING}{all} (no parameters)

{P|PARAGRAPH}<n1><n2><n3>{all}

{PR|PERIOD}{all} (no parameters)

{PX|PRINT@INDEX}{all} (no parameters)

{RPT|REPEAT}<q>{all}

{REQ|REQUIRE}<q>{all}

{R|RIGHT}<n>{;|EL}<text1>{EL}

{RM|RIGHT@MARGIN}<n>{all}

{STC|SEND@TOC}<n><text1>{all}

{SDT|SET@DATE}<n1><n2><n3>{all}

{SPR|SET@PARAGRAPH}<n1><n2><n3>{all}

{STM|SET@TIME}<n1><n2><n3>{all}

{S|SKIP}<n>{all}

{SP|SPACING}<n>{all}

{SD|STANDARD}<n>{all}

{STHL|STYLE@HEADERS}<n1><n2><n3>{all}

{IX|SUBINDEX}<text1>{all}

{SPG|SUBPAGE}{all} (no parameters)

{ST|SUBTTL"|SUBTITLE}<text1>{all}

{TS|TAB@STOPS}<n1><n2>...<n32>{all}

{TP|TEST@PAGE}<n>{all}

{T|TITLE}<text1>{all}

{UC|UPPER@CASE}{all} (no parameters)

{VR|VARIABLE}<name><flags>{all}

APPENDIX D

TEMPLATE SAMPLES

D.1 MEMO TEMPLATE

The following provides an example of a memorandum template and the manner in which it was derived.

I N T E R O F F I C E M E M O R A N D U M

To: R. Friday

Date: 8 October 1979
From: Art Ellis
Dept: BLISS Development
Ext: DTN 493-8684
Loc/Mail Stop: ML3-5/E82
File:

Subject: DSR Memo Template

(The text is input here.)

Art Ellis

Memo Template Formatting

The following depicts the formatting of the memo template.

```
.ps60,70
.flag bold
.s;^*+-------------+
.br;| | | | | | | |
.br;|d|i|g|i|t|a|l|######I N T E R O F F I C E####M E M O R A N D U M
.br;| | | | | | | |
.br;+-------------+\*
.no flag bold
.ts39,53 .lm39
.no period
.s2.i-39;To:#R.#Friday
.period
.flag substitute
                                        Date:##$$day $$month $$year
.br;From:#Art#Ellis
.br;Dept: BLISS Development
.br;Ext:###DTN 493-8684
.br;Loc/Mail Stop: ML3-5/E82
.br;File:
.ts20.lm0.s3;Subject:  DSR Memo Template
.title Creating a Template
.b2.lm4.rm64
(The text is input here.)
.b4.ts50;                               (signature)
```

D.2 REPORT TEMPLATE

The following provides an example of a report template and the manner in which it was derived.

```
+---------------+
! d i g i t a l !  I n t e r o f f i c e  M e m o r a n d u m
+---------------+

To:  List                          Date:  8 October 1979
                                   From:  A. F. Ellis
                                   Dept:  Software Methodology
                                   Ext.:  7838 Loc.:  ML3-5/E82
                                   File:  <MMMYY>.RNO
                                   Memo:  not used     Rev.:  0
 List: R.W. Friday
       C. Bradley
       W. Clark
 cc:

Subj.:  Report for October, 1979

1.0  Accomplishments
2.0  Not Accomplished
3.0  Scheduled for Next Month
4.0  Problems
5.0  Solutions
6.0  Miscellaneous

[end of <MMMYY>.RNO]
```

Report Template Format

The following depicts the formatting of the report template.

```
.flag substitute
.title Subj.:#Report for $$month, $$year
.skip2.lm0.nf.ts13,25,38,50,63
.spl.pg.rm65
+---------------+
! d i g i t a l !   I n t e r o f f i c e  M e m o r a n d u m
+---------------+
.skip2.ts30
To:##List       Date:##$$day $$month $$year
        From:##A.#F.#Ellis
        Dept:##Software Methodology
        Ext.:##7838#Loc.:##ML3-5/E82
        File:##<MMMYY>.RNO
        Memo:##not used#####Rev.:##0
.lm 7.i-6
List:#R.#W.#Friday
W.#White
J.#Jones
.lm 5.i-4
cc:#
.lm0.ts 8,16,24,32,40,48,56,64,72
.skip2.f.lm11.i-11;Subj.:##Report for $$month, $$year
.skip2.lm0
.hl 1 Accomplishments
.hl 1 Not Accomplished
.hl 1 Scheduled for Next Month
.hl 1 Problems
.hl 1 Solutions
.hl 1 Miscellaneous
.lm0.skip2;[end of#<MMMYY>.RNO]
```

INDEX

DEC Standard Runoff
User's Guide
AA-J268A-TK

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
or
Country

—Do Not Tear - Fold Here and Tape —

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE COMMUNICATIONS
P.O. Box F
Maynard, Massachusetts 01754

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line

digital

digital